Istruzioni generali per controllori Logix5000

Manuale di Riferimento

Numeri di catalogo 1756-L1x, 1756-L5x, 1756-L6x, 1768-L4x, 1769-L30, 1769-L31, 1769-L32C, 1769-L32E, 1769-L35CR, 1769-L35E, 1789-L60, 1794-L34, 20D

ALLEN-BRADLEY • ROCKWELL SOFTWARE

Rockwell Automation

Informazioni importanti per l'utente

Le apparecchiature a stato solido hanno caratteristiche di funzionamento diverse da quelle delle apparecchiature elettromeccaniche. Nel manuale Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (pubblicazione SGI-1.1 disponibile presso l'ufficio vendite locale di Rockwell Automation oppure online all'indirizzo http://literature.rockwellautomation.com) sono descritte alcune differenze importanti tra le apparecchiature a stato solido e i dispositivi elettromeccanici cablati. A causa di questa differenza e della grande varietà di utilizzo delle apparecchiature a stato solido, tutte le persone responsabili dell'applicazione di questa apparecchiatura devono assicurarsi che ogni applicazione di questa apparecchiatura sia accettabile.

In nessun caso Rockwell Automation, Inc. sarà responsabile per danni indiretti derivanti dall'utilizzo o dall'applicazione di questa apparecchiatura.

Gli esempi gli schemi contenuti nel presente manuale sono inclusi esclusivamente a scopo illustrativo. Poiché le variabili e i requisiti associati alle installazioni specifiche sono innumerevoli, Rockwell Automation, Inc. non può essere ritenuta responsabile per l'utilizzo effettivo basato sugli esempi e sui diagrammi qui riportati.

Rockwell Automation, Inc. declina qualsiasi responsabilità brevettuale in relazione all'utilizzo di informazioni, circuiti, apparecchiatura o software descritti nel presente manuale.

La riproduzione totale o parziale del contenuto del presente manuale è vietata senza il consenso scritto di Rockwell Automation, Inc.

Nel presente manuale si è fatto ricorso all'uso di note per illustrare all'utente le considerazioni in materia di sicurezza.

AVVISO	Identifica informazioni sulle pratiche o le circostanze che possono causare un'esplosione in un ambiente pericoloso, generando infortuni al personale o decesso, danni alle proprietà o perdite economiche.
IMPORTANTE	Identifica informazioni importanti per la buona applicazione e comprensione del prodotto.
ATTENZIONE	Identifica informazioni sulle pratiche o le circostanze che possono causare lesioni al personale o decesso, danni alle cose o perdite economiche. Le indicazioni aiuteranno l'utente ad individuare i rischi, evitarli e prevederne le conseguenze.
PERICOLO DI SCOSSE ELETTRICHE	Potranno essere collocate delle etichette sull'apparecchiatura o al suo interno, per esempio su azionamento o motore, per attirare l'attenzione dell'utente sulla tensione potenzialmente pericolosa presente.
PERICOLO DI USTIONI	Potranno essere collocate delle etichette sull'apparecchiatura o al suo interno, per esempio su azionamento o motore, per attirare l'attenzione dell'utente sulle superfici che potrebbero raggiungere temperature potenzialmente pericolose.

Allen-Bradley, Rockwell Automation e TechConnect sono marchi commerciali di Rockwell Automation, Inc.

I marchi commerciali non di proprietà di Rockwell Automation sono di proprietà delle rispettive aziende titolari.

Prefazione

Istruzioni generali dei controllori Logix5000

Introduzione 23
A chi è rivolto questo manuale 23
Scopo di questo manuale 24
Informazioni comuni per tutte le istruzioni 25
Convenzioni e termini correlati 25
Impostato e azzerato 25
Condizione del ramo in logica ladder 26
Stati dei blocchi funzione 27

Capitolo 1

Istruzioni di allarmi ed eventi FactoryTalk basati su Logix (ALMD, ALMA)

Introduzione 29
Digital Alarm (ALMD) 30
Diagrammi di stato quando è richiesta la conferma di ricezione 36
Diagrammi di stato quando la conferma della ricezione non è richiesta 37
Allarme ALMD, conferma di ricezione richiesta e agganciato 38
Allarme ALMD, conferma di ricezione richiesta e non agganciato 39
Allarme ALMD, conferma di ricezione non richiesta e agganciato 39
Allarme ALMD, conferma di ricezione non richiesta e non agganciato 40
Analog Alarm (ALMA) 42
Diagrammi di stato quando è richiesta la conferma di ricezione 54
Diagrammi di stato quando non è richiesta la conferma di ricezione 55
Condizione livello ALMA – Conferma di ricezione richiesta 57
Condizione livello ALMA – Conferma di ricezione non richiesta 58
Tasso di variazione ALMA – Conferma di ricezione richiesta 59
Tasso di variazione ALMA – Conferma di ricezione non richiesta 60
Configurazione di un'istruzione di allarme 61
Immissione del testo di un messaggio di allarme 64
Variabili della stringa del messaggio 65
Versioni in più lingue dei messaggi di allarme 66
Monitoraggio dello stato degli allarmi 67
Bufferizzazione degli allarmi 67
Accesso via programma alle informazioni di allarme 68
Soppressione o disabilitazione degli allarmi 70
Esecuzione degli allarmi del controllore 71
Utilizzo della memoria del controllore 72
Tempo di scansione 73

Capitolo 2

Istruzioni di bit (XIC, XIO, OTE, OTL, OTU, ONS, OSR, OSF, OSRI, OSFI)

Introduzione 75
Examine If Closed (XIC) 76
Examine If Open (XIO) 78
Output Energize (OTE) 80
Output Latch (OTL) 82
Output Unlatch (OTU) 84
One Shot (ONS) 86
One Shot Rising (OSR) 89
One Shot Falling (OSF) 92
One Shot Rising with Input (OSRI) 94
One Shot Falling with Input (OSFI) 97

Capitolo 3

Istruzioni di timer e contatori (TON, TOF, RTO, TONR, TOFR, RTOR, CTU, CTD, CTUD, RES)

Introduzione 101
Timer On Delay (TON) 102
Timer Off Delay (TOF) 106
Retentive Timer On (RTO) 110
Timer On Delay with Reset (TONR) 114
Timer Off Delay with Reset (TOFR) 118
Retentive Timer On with Reset (RTOR) 122
Count Up (CTU) 126
Count Down (CTD) 130
Count Up/Down (CTUD) 134
Reset (RES) 139

Capitolo 4

Istruzioni di I/O (MSG, GSV, SSV, IOT)

Introduzione 141
Message (MSG) 142
Codici di errore MSG 150
    Codici di errore 150
    Codici di errore estesi 151
    Codici di errore di PLC e SLC (.ERR) 153
    Codici di errore dei trasferimenti a blocchi 155
Specificare i dettagli di configurazione 156
    Specificare la Tabella dati CIP per messaggi di lettura e scrittura 157
    Riconfigurare un modulo di I/O 158
    Specificare messaggi CIP generici 159
    Specificare messaggi per PLC-5 160
    Specificare messaggi per SLC 162
    Specificare messaggi di trasferimento a blocchi 162
    Specificare messaggi per PLC-3 163
    Specificare messaggi per PLC-2 164

Esempi di configurazione MSG 165
Specificare i dettagli di comunicazione. 166
Specificare un percorso 166
Per trasferimenti a blocchi 169
Specificare un metodo di comunicazione o l'indirizzo di un modulo 169
Scegliere un'opzione cache 170
Regole generali 172
Get System Value (GSV) and Set System Value (SSV) 173
Oggetti GSV/SSV. 176
Accesso all'oggetto CONTROLLER 177
Accesso all'oggetto CONTROLLERDEVICE 178
Accesso all'oggetto CST 179
Accesso all'oggetto DF1 181
Accesso all'oggetto FAULTLOG. 184
Accesso all'oggetto MESSAGE. 185
Accesso all'oggetto MODULE 187
Accesso all'oggetto MOTIONGROUP 188
Accesso all'oggetto PROGRAM 189
Accesso all'oggetto Routine 190
Accesso all'oggetto SERIALPORT 190
Accesso all'oggetto TASK 192
Accesso all'oggetto WALLCLOCKTIME 194
Esempio di programmazione GSV/SSV. 195
Get Fault Information. 195
Impostazione degli indicatori di abilitazione e disabilitazione 197
Immediate Output (IOT) 198

Capitolo 5

Istruzioni di confronto (CMP, EQU, GEQ, GRT, LEQ, LES, LIM, MEQ, NEQ)

Introduzione. 203
Compare (CMP) 204
Espressioni CMP 206
Operatori validi 206
Formattazione delle espressioni 206
Determinazione dell'ordine delle operazioni 207
Uso di stringhe in un'espressione 208
Equal to (EQU). 209
Greater than or Equal to (GEQ). 213
Greater Than (GRT) 217
Less Than or Equal to (LEQ) 221
Less Than (LES) 225
Limit (LIM) 229
Mask Equal to (MEQ) 235
Immissione di un valore maschera immediato. 236
Not Equal to (NEQ). 240

Capitolo 6

Istruzioni di calcolo/matematiche (CPT, ADD, SUB, MUL, DIV, MOD, SQR, SQRT, NEG, ABS)

Introduzione 245
Compute (CPT) 246
- Operatori validi 248
- Formattazione delle espressioni 248
- Determinazione dell'ordine delle operazioni 249

Add (ADD) 250
Subtract (SUB) 253
Multiply (MUL) 256
Divide (DIV) 259
Modulo (MOD) 264
Square Root (SQR) 268
Negate (NEG) 272
Absolute Value (ABS) 275

Capitolo 7

Istruzioni di spostamento/logiche (MOV, MVM, BTD, MVMT, BTDT, CLR, SWPB, AND, OR, XOR, NOT, BAND, BOR, BXOR, BNOT)

Introduzione 279
Move (MOV) 281
Masked Move (MVM) 283
- Immissione di un valore di maschera immediato 284

Masked Move with Target (MVMT) 286
Bit Field Distribute (BTD) 290
Bit Field Distribute with Target (BTDT) 293
Clear (CLR) 296
Swap Byte (SWPB) 298
Bitwise AND (AND) 302
Bitwise OR (OR) 305
Bitwise Exclusive OR (XOR) 308
Bitwise NOT (NOT) 311
Boolean AND (BAND) 314
Boolean OR (BOR) 317
Boolean Exclusive OR (BXOR) 320
Boolean NOT (BNOT) 323

Capitolo 8

Istruzioni di matrice (file)/varie (FAL, FSC, COP, CPS, FLL, AVE, SRT, STD, SIZE)

Introduzione 325
Scelta della modalità operativa 326
- Modalità Continua 326
- Modalità discreta 327
- Modalità incrementale 329

File Arithmetic and Logic (FAL) 331
- Espressioni FAL 340
- Operatori validi 340
- Formattazione delle espressioni 341
- Determinazione dell'ordine delle operazioni 341

File Search and Compare (FSC) 342
Espressioni FSC 346
Operatori validi 347
Formattazione delle espressioni 347
Determinazione dell'ordine delle operazioni 348
Uso di stringhe in un'espressione 349
Copy File (COP) Synchronous Copy File (CPS). 350
File Fill (FLL) 356
File Average (AVE) 360
File Sort (SRT). 365
File Standard Deviation (STD) 370
Size In Elements (SIZE) 375

Capitolo 9

Istruzioni di matrici (File)/Scorrimento (BSL, BSR, FFL, FFU, LFL, LFU)

Introduzione 379
Bit Shift Left (BSL). 380
Bit Shift Right (BSR) 384
FIFO Load (FFL) 388
FIFO Unload (FFU) 394
LIFO Load (LFL) 400
LIFO Unload (LFU) 406

Capitolo 10

Istruzioni sequenziatore (SQI, SQO, SQL)

Introduzione 413
Sequencer Input (SQI). 414
Immissione di un valore di maschera immediato 415
Uso di SQI senza SQO 417
Sequencer Output (SQO) 418
Immissione di un valore di maschera immediato 419
Uso di SQI con SQO 421
Ripristino della posizione dell'istruzione SQO 421
Sequencer Load (SQL). 422

Capitolo 11

Istruzioni di controllo programma (JMP, LBL, JSR, RET, SBR, JXR, TND, MCR, UID, UIE, AFI, NOP, EOT, SFP, SFR, EVENT)

Introduzione 427
Jump to Label (JMP) Label (LBL) 428
Jump to Subroutine (JSR) Subroutine (SBR) Return (RET). 430
Jump to External Routine (JXR) 441
Temporary End (TND) 444
Master Control Reset (MCR). 446
User Interrupt Disable (UID) User Interrupt Enable (UIE). 448

Always False Instruction (AFI) 450
No Operation (NOP) 451
End of Transition (EOT) 452
SFC Pause (SFP) 454
SFC Reset (SFR) 456
Trigger Event Task (EVENT) 458
Determinare via programma se un'istruzione EVENT ha azionato un task 458

Capitolo 12

Istruzioni For/Break (FOR, FOR...DO, BRK, EXIT, RET)

Introduzione 463
For (FOR) 464
Break (BRK) 467
Return (RET) 468

Capitolo 13

Istruzioni speciali (FBC, DDT, DTR, PID)

Introduzione 471
File Bit Comparison (FBC) 472
Scelta della modalità di ricerca 474
Diagnostic Detect (DDT) 479
Scelta della modalità di ricerca 481
Data Transitional (DTR) 486
Immissione di un valore di maschera immediato 487
Proportional Integral Derivative (PID) 489
Configurazione di un'istruzione PID 494
Messa a punto (Tuning) 495
Configurazione 495
Allarmi 496
Conversione in scala 497
Uso delle istruzioni PID 497
Antiaccumulo dell'integrale e trasferimento bumpless da manuale ad automatico 499
Temporizzazione dell'istruzione PID 500
Riavviamento bumpless 504
Attenuazione derivata 505
Impostazione della banda morta 506
Uso del limite dell'uscita 506
Compensazione anticipata o bias dell'uscita 507
Anelli in cascata 507
Controllo di un rapporto 508
Teoria PID 509
Processo PID 509
Processo PID con anelli master/slave 509

Capitolo 14

Istruzioni trigonometriche (SIN, COS, TAN, ASN, ASIN, ACS, ACOS, ATN, ATAN)

Introduzione 511
Sine (SIN) 512
Cosine (COS) 515
Tangent (TAN) 518
Arc Sine (ASN) 521
Arc Cosine (ACS) 524
Arc Tangent (ATN) 527

Capitolo 15

Istruzioni di matematica avanzata (LN, LOG, XPY)

Introduzione 531
Natural Log (LN) 532
Log Base 10 (LOG) 535
X to the Power of Y (XPY) 538

Capitolo 16

Istruzioni di conversione matematica (DEG, RAD, TOD, FRD, TRN, TRUNC)

Introduzione 541
Degrees (DEG) 542
Radians (RAD) 545
Convert to BCD (TOD) 548
Convert to Integer (FRD) 551
Truncate (TRN) 553

Capitolo 17

Istruzioni Porta Seriale ASCII (ABL, ACB, ACL, AHL, ARD, ARL, AWA, AWT)

Introduzione 557
Esecuzione delle istruzioni 558
Codici di errore ASCII 560
Tipi di dati stringa 560
ASCII Test For Buffer Line (ABL) 562
ASCII Chars in Buffer (ACB) 565
ASCII Clear Buffer (ACL) 567
ASCII Handshake Lines (AHL) 569
ASCII Read (ARD) 573
ASCII Read Line (ARL) 577
ASCII Write Append (AWA) 581
ASCII Write (AWT) 586

Capitolo 18

Istruzioni di stringhe ASCII (CONCAT, DELETE, FIND, INSERT, MID)

Introduzione 591
Tipi di dati stringa 592
String Concatenate (CONCAT) 593
String Delete (DELETE) 595
Find String (FIND) 597
Insert String (INSERT) 599
Middle String (MID) 601

Capitolo 19

Istruzioni di conversione ASCII (STOD, STOR, DTOS, RTOS, UPPER, LOWER)

Introduzione 603
Tipi di dati stringa 605
String To DINT (STOD) 606
String To REAL (STOR) 608
DINT to String (DTOS) 611
REAL to String (RTOS) 613
Upper Case (UPPER) 615
Lower Case (LOWER) 617

Capitolo 20

Istruzioni di debug (BPT, TPT)

Introduzione 619
Breakpoints (BPT) 619
Stringa Format 620
Tracepoints (TPT) 623
Stringa Format 623

Appendice A

Attributi comuni

Introduzione 627
Valori immediati 627
Conversioni di dati 627
Da SINT o INT a DINT 629
Da intero a REAL 631
Da DINT a SINT o INT 631
Da REAL a intero 632

Appendice B

Attributi di un blocco funzione

Introduzione 633
Selezione degli elementi del blocco funzione 633
Aggancio dati 634
Ordine di esecuzione 636
Risoluzione di un loop 637
Risoluzione del flusso di dati tra due blocchi 638
Creazione di un ritardo di una scansione 639
Riepilogo 639
Risposte dei blocchi funzione a condizioni di overflow 640
Modalità di temporizzazione 641
Parametri di istruzione comuni per le modalità di temporizzazione 643
Prospetto delle modalità di temporizzazione 645
Controllo Programma/Operatore 646

Appendice C

Programmazione in testo strutturato

Introduzione 651
Sintassi del testo strutturato 651
Assegnazione 653
Specificare un'assegnazione non ritentiva 654
Assegnare un carattere ASCII a una stringa 655
Espressioni 655
Usare funzioni e operatori aritmetici 657
Usare operatori relazionali 658
Usare operatori logici 660
Utilizzo di operatori bit per bit 661
Determinare l'ordine di esecuzione 661
Istruzioni 662
Costrutti 663
Alcune parole chiave sono riservate all'uso futuro 663
IF...THEN 664
CASE...OF 667
FOR…DO 670
WHILE…DO 673
REPEAT…UNTIL 676
Commenti 679

Indice analitico

Codici dei caratteri ASCII 691

Introduzione

La presente versione del documento contiene informazioni nuove e aggiornate.

Informazioni aggiornate

Questo documento contiene le seguenti modifiche:

Modifica	Pagina
Capitolo 1 – Istruzioni Allarme digitale (Digital Alarm) (ALMD) e Allarme analogico (Analog Alarm) (ALMA) in un unico capitolo. Aggiunte informazioni su configurazione, stringhe di messaggio e stato.	29
Capitolo 4 – Aggiunti nuovi attributi dell'oggetto controllore GSV/SSV.	177
Capitolo 20 – Aggiunte informazioni per il debug (PPT, TPT).	619

Note:

Dove trovare un'istruzione

Questo indice serve per trovare informazioni di riferimento sulle istruzioni Logix (le istruzioni con sfondo grigio sono disponibili in altri manuali). Vengono riportati anche i linguaggi di programmazione delle istruzioni.

Se viene riportato	L'istruzione viene descritta in
un numero di pagina	questo manuale
movimento	*Logix5000 Set di istruzioni di movimento per controllori Logix5000 – Manuale di riferimento,* pubblicazione 1756-RM007
PhaseManager	*Manuale per l'utente di PhaseManager*, pubblicazione LOGIX-UM001
controllo di processo	*Manuale di riferimento delle istruzioni per controllo di processo ed azionamenti per controllori Logix5000*, pubblicazione 1756-RM006

Istruzione	Riferimento	Linguaggi
ABL ASCII Test For Buffer Line	562	ladder testo strutturato
ABS Absolute Value	275	ladder testo strutturato blocco funzione
ACB ASCII Chars in Buffer	565	ladder testo strutturato
ACL ASCII Clear Buffer	567	ladder testo strutturato
ACS Arc Cosine	524	ladder testo strutturato blocco funzione
ADD Add	250	ladder testo strutturato blocco funzione
AFI Always False Instruction	450	ladder
AHL ASCII Handshake Lines	569	ladder testo strutturato
ALM Alarm	controllo di processo	testo strutturato blocco funzione
ALMA Analog Alarm	42	ladder testo strutturato blocco funzione
ALMD Digital Alarm	30	ladder testo strutturato blocco funzione
AND Bitwise AND	302	ladder testo strutturato blocco funzione
ARD ASCII Read	573	ladder testo strutturato
ARL ASCII Read Line	577	logica ladder testo strutturato
ASN Arc Sine	521	ladder testo strutturato blocco funzione
ATN Arc Tangent	527	ladder testo strutturato blocco funzione
AVE File Average	360	ladder
AWA ASCII Write Append	581	ladder testo strutturato
AWT ASCII Write	586	ladder testo strutturato
BAND Boolean AND	314	testo strutturato blocco funzione
BNOT Boolean NOT	323	testo strutturato blocco funzione
BOR Boolean OR	317	testo strutturato blocco funzione

Istruzione	Riferimento	Linguaggi
BPT Breakpoints	619	ladder
BRK Break	467	ladder
BSL Bit Shift Left	380	ladder
BSR Bit Shift Right	384	ladder
BTD Bit Field Distribute	290	ladder
BTDT Bit Field Distribute with Target	293	testo strutturato blocco funzione
BTR Message	142	ladder testo strutturato
BTW Message	142	ladder testo strutturato
BXOR Boolean Exclusive OR	320	testo strutturato blocco funzione
CC Coordinated Control	controllo di processo	testo strutturato blocco funzione
CLR Clear	296	ladder testo strutturato
CMP Compare	204	ladder
CONCAT String Concatenate	593	ladder testo strutturato
COP Copy File	350	ladder testo strutturato
COS Cosine	515	ladder testo strutturato blocco funzione
CPS Synchronous Copy File	350	ladder testo strutturato
CPT Compute	246	ladder
CTD Count Down	130	ladder
CTU Count Up	126	ladder
CTUD Count Up/Down	134	testo strutturato blocco funzione
D2SD Discrete 2-State Device	controllo di processo	testo strutturato blocco funzione
D3SD Discrete 3-State Device	controllo di processo	testo strutturato blocco funzione
DDT Diagnostic Detect	479	ladder

Istruzione	Riferimento	Linguaggi
DEDT Deadtime	controllo di processo	testo strutturato blocco funzione
DEG Degrees	542	ladder testo strutturato blocco funzione
DELETE String Delete	595	ladder testo strutturato
DERV Derivative	controllo di processo	testo strutturato blocco funzione
DFF D Flip-Flop	controllo di processo	testo strutturato blocco funzione
DIV Divide	259	ladder testo strutturato blocco funzione
DTOS DINT to String	611	ladder testo strutturato
DTR Data Transitional	486	ladder
EOT End of Transition	452	ladder testo strutturato
EQU Equal to	209	ladder testo strutturato blocco funzione
ESEL Enhanced Select	controllo di processo	testo strutturato blocco funzione
EVENT Trigger Event Task	458	ladder testo strutturato
FAL File Arithmetic and Logic	331	ladder
FBC File Bit Comparison	472	ladder
FFL FIFO Load	388	ladder
FFU FIFO Unload	394	ladder
FGEN Function Generator	controllo di processo	testo strutturato blocco funzione
FIND Find String	597	ladder testo strutturato
FLL File Fill	356	ladder
FOR FOR	464	ladder
FRD Convert to Integer	551	ladder blocco funzione
FSC File Search and Compare	342	ladder
GEQ Greater than or Equal to	213	ladder testo strutturato blocco funzione
GRT Greater Than	217	ladder testo strutturato blocco funzione
GSV Get System Value	173	ladder testo strutturato
HLL High/Low Limit	controllo di processo	testo strutturato blocco funzione
HPF High Pass Filter	controllo di processo	testo strutturato blocco funzione
ICON Input Wire Connector	633	blocco funzione
IMC Internal Model Control	controllo di processo	testo strutturato blocco funzione
INSERT Insert String	599	ladder testo strutturato
INTG Integrator	controllo di processo	testo strutturato blocco funzione
IOT Immediate Output	198	ladder testo strutturato
IREF Input Reference	633	blocco funzione
JKFF JK Flip-Flop	controllo di processo	testo strutturato blocco funzione
JMP Jump to Label	428	ladder
JSR Jump to Subroutine	430	ladder testo strutturato blocco funzione
JXR Jump to External Routine	441	ladder
LBL Label	428	ladder
LDL2 Second-Order Lead Lag	controllo di processo	testo strutturato blocco funzione
LDLG Lead-Lag	controllo di processo	testo strutturato blocco funzione
LEQ Less Than or Equal to	221	ladder testo strutturato blocco funzione
LES Less Than	225	ladder testo strutturato blocco funzione
LFL LIFO Load	400	ladder
LFU LIFO Unload	406	ladder
LIM Limit	229	ladder blocco funzione

Istruzione	Riferimento	Linguaggi
LN Natural Log	532	ladder testo strutturato blocco funzione
LOG Log Base 10	(1)	ladder testo strutturato blocco funzione
LOWER Lower Case	617	ladder testo strutturato
LPF Low Pass Filter	controllo di processo	testo strutturato blocco funzione
MAAT Motion Apply Axis Tuning	movimento	ladder testo strutturato
MAFR Motion Axis Fault Reset	movimento	ladder testo strutturato
MAG Motion Axis Gear	movimento	ladder testo strutturato
MAHD Motion Apply Hookup Diagnostics	movimento	ladder testo strutturato
MAH Motion Axis Home	movimento	ladder testo strutturato
MAJ Motion Axis Jog	movimento	ladder testo strutturato
MAM Motion Axis Move	movimento	ladder testo strutturato
MAOC Motion Arm Output Cam	movimento	ladder testo strutturato
MAPC Motion Axis Position Cam	movimento	ladder testo strutturato
MAR Motion Arm Registration	movimento	ladder testo strutturato
MASD Motion Axis Shutdown	movimento	ladder testo strutturato
MAS Motion Axis Stop	movimento	ladder testo strutturato
MASR Motion Axis Shutdown Reset	movimento	ladder testo strutturato
MATC Motion Axis Time Cam	movimento	ladder testo strutturato
MAVE Moving Average	controllo di processo	testo strutturato blocco funzione
MAW Motion Arm Watch	movimento	ladder testo strutturato
MAXC Maximum Capture	controllo di processo	testo strutturato blocco funzione
MCCD Motion Coordinated Change Dynamics	movimento	ladder testo strutturato
MCCM Motion Coordinated Circular Move	movimento	ladder testo strutturato
MCCP Motion Calculate Cam Profile	movimento	ladder testo strutturato
MCD Motion Change Dynamics	movimento	ladder testo strutturato
MCLM Motion Coordinated Linear Move	movimento	ladder testo strutturato
MCR Master Control Reset	446	ladder
MCSD Motion Coordinated Shutdown	movimento	ladder testo strutturato
MCS Motion Coordinated Stop	movimento	ladder testo strutturato
MCSR Motion Coordinated Shutdown Reset	movimento	ladder testo strutturato
MCT Motion Coordinated Transform	movimento	ladder testo strutturato
MCTP Motion Calculate Transform Position	movimento	ladder testo strutturato
MDF Motion Direct Drive Off	movimento	ladder testo strutturato
MDOC Motion Disarm Output Cam	movimento	ladder testo strutturato
MDO Motion Direct Drive On	movimento	ladder testo strutturato
MDR Motion Disarm Registration	movimento	ladder testo strutturato
MDW Motion Disarm Watch	movimento	ladder testo strutturato
MEQ Mask Equal to	235	ladder testo strutturato blocco funzione
MGSD Motion Group Shutdown	movimento	ladder testo strutturato
MGS Motion Group Stop	movimento	ladder testo strutturato
MGSP Motion Group Strobe Position	movimento	ladder testo strutturato
MGSR Motion Group Shutdown Reset	movimento	ladder testo strutturato
MID Middle String	601	ladder testo strutturato

Istruzione	Riferimento	Linguaggi
MINC Minimum Capture	controllo di processo	testo strutturato blocco funzione
IMMC Modular Multivariable Control	controllo di processo	testo strutturato blocco funzione
MOD Modulo	264	ladder testo strutturato blocco funzione
MOV Move	281	ladder
MRAT Motion Run Axis Tuning	movimento	ladder testo strutturato
MRHD Motion Run Hookup Diagnostics	movimento	ladder testo strutturato
MRP Motion Redefine Position	movimento	ladder testo strutturato
MSF Motion Servo Off	movimento	ladder testo strutturato
MSG Message	142	ladder testo strutturato
MSO Motion Servo On	movimento	ladder testo strutturato
MSTD Moving Standard Deviation	controllo di processo	testo strutturato blocco funzione
MUL Multiply	256	ladder testo strutturato blocco funzione
MUX Multiplexer	controllo di processo	blocco funzione
MVM Masked Move	283	ladder
MVMT Masked Move with Target	286	testo strutturato blocco funzione
NEG Negate	272	ladder testo strutturato blocco funzione
NEQ Not Equal to	240	ladder testo strutturato blocco funzione
NOP No Operation	451	ladder
NOT Bitwise NOT	311	ladder testo strutturato blocco funzione
NTCH Notch Filter	controllo di processo	testo strutturato blocco funzione
OCON Output Wire Connector	633	blocco funzione
ONS One Shot	86	ladder
OR Bitwise OR	305	ladder testo strutturato blocco funzione
OREF Output Reference	633	blocco funzione
OSFI One Shot Falling with Input	97	testo strutturato blocco funzione
OSF One Shot Falling	92	ladder
OSRI One Shot Rising with Input	94	testo strutturato blocco funzione
OSR One Shot Rising	89	ladder
OTE Output Energize	80	ladder
OTL Output Latch	82	ladder
OTU Output Unlatch	84	ladder
PATT Attach to Equipment Phase	PhaseManager	ladder testo strutturato
PCLF Equipment Phase Clear Failure	PhaseManager	ladder testo strutturato
PCMD Equipment Phase Command	PhaseManager	ladder testo strutturato
PDET Detach from Equipment Phase	PhaseManager	ladder testo strutturato
PFL Equipment Phase Failure	PhaseManager	ladder testo strutturato
PIDE Enhanced PID	controllo di processo	testo strutturato blocco funzione
PID Proportional Integral Derivative	489	ladder testo strutturato
PI Proportional + Integral	controllo di processo	testo strutturato blocco funzione
PMUL Pulse Multiplier	controllo di processo	testo strutturato blocco funzione
POSP Position Proportional	controllo di processo	testo strutturato blocco funzione
POVR Equipment Phase Override Command	PhaseManager	ladder testo strutturato
PPD Equipment Phase Paused	PhaseManager	ladder testo strutturato
PRNP Equipment Phase New Parameters	PhaseManager	ladder testo strutturato

Istruzione	Riferimento	Linguaggi
PSC Phase State Complete	PhaseManager	ladder testo strutturato
PXRQ Equipment Phase External Request	PhaseManager	ladder testo strutturato
RAD Radians	545	ladder testo strutturato blocco funzione
RESD Reset Dominant	controllo di processo	testo strutturato blocco funzione
RES Reset	139	ladder
RET Return	430 e 468	ladder testo strutturato blocco funzione
RLIM Rate Limiter	controllo di processo	testo strutturato blocco funzione
RMPS Ramp/Soak	controllo di processo	testo strutturato blocco funzione
RTO Retentive Timer On	110	ladder
RTOR Retentive Timer On with Reset	122	testo strutturato blocco funzione
RTOS REAL to String	613	ladder testo strutturato
SBR Subroutine	430	ladder testo strutturato blocco funzione
SCL Scale	controllo di processo	testo strutturato blocco funzione
SCRV S-Curve	controllo di processo	testo strutturato blocco funzione
SEL Select	controllo di processo	blocco funzione
SETD Set Dominant	controllo di processo	testo strutturato blocco funzione
SFP SFC Pause	454	ladder testo strutturato
SFR SFC Reset	456	ladder testo strutturato
SIN Sine	512	ladder testo strutturato blocco funzione
SIZE Size In Elements	375	ladder testo strutturato
SNEG Selected Negate	controllo di processo	testo strutturato blocco funzione
SOC Second-Order Controller	controllo di processo	testo strutturato blocco funzione
SQI Sequencer Input	414	ladder
SQL Sequencer Load	422	ladder
SQO Sequencer Output	418	ladder
SQR Square Root	268	ladder blocco funzione
SQRT Square Root	268	testo strutturato
SRT File Sort	365	ladder testo strutturato
SRTP Split Range Time Proportional	controllo di processo	testo strutturato blocco funzione
SSUM Selected Summer	controllo di processo	testo strutturato blocco funzione
SSV Set System Value	173	ladder testo strutturato
STD File Standard Deviation	370	ladder
STOD String To DINT	606	ladder testo strutturato
STOR String To REAL	608	ladder testo strutturato
SUB Subtract	253	ladder testo strutturato blocco funzione
SWPB Swap Byte	298	ladder testo strutturato
TAN Tangent	518	ladder testo strutturato blocco funzione
TND Temporary End	444	ladder
TOD Convert to BCD	548	ladder blocco funzione
TOFR Timer Off Delay with Reset	118	testo strutturato blocco funzione
TOF Timer Off Delay	106	ladder
TONR Timer On Delay with Reset	114	testo strutturato blocco funzione
TON Timer on Delay	102	ladder
TOT Totalizer	controllo di processo	testo strutturato blocco funzione
TPT Tracepoints	623	ladder

Istruzione	Riferimento	Linguaggi
TRN Truncate	553	ladder blocco funzione
TRUNC Truncate	553	testo strutturato
UID User Interrupt Disable	448	ladder testo strutturato
UIE User Interrupt Enable	448	ladder testo strutturato
UPDN Up/Down Accumulator	controllo di processo	testo strutturato blocco funzione
UPPER Upper Case	615	ladder testo strutturato
XIC Examine If Closed	76	ladder
XIO Examine if Open	78	ladder
XOR Bitwise Exclusive OR	308	ladder testo strutturato blocco funzione
XPY X to the Power of Y	538	ladder testo strutturato blocco funzione

Notizen:

Istruzioni generali dei controllori Logix5000

Introduzione

Questo manuale è solo uno dei vari documenti contenenti istruzioni relative ai controllori Logix5000.

Obiettivo/scopo	Documenti
Programmazione del controllore per applicazioni sequenziali **Voi siete qui**	*Controllori Logix5000 Set di istruzioni generali – Manuale di riferimento*, pubblicazione 1756-RM003
Programmazione del controllore per applicazioni di processo o azionamenti	*Manuale di riferimento con istruzioni per controllo di processo ed azionamenti per controllori Logix5000*, pubblicazione 1756-RM006
Programmazione del controllore per applicazioni di movimento	*Logix5000 Set di istruzioni di movimento per controllori Logix5000 – Manuale di riferimento*, pubblicazione 1756-RM007
Programmazione del controllore per usare le fasi dell'attrezzatura	*Manuale per l'utente di PhaseManager,* pubblicazione LOGIX-UM001
Importazione di un file di testo o di tag in un progetto	*Importazione/esportazione con il Controllore Logix5000 – Manuale di riferimento* , pubblicazione 1756-RM084
Esportazione di un progetto o di tag in un file di testo	
Conversione di un'applicazione PLC-5 o SLC 500 in un'applicazione Logix5000	*Conversione dalla logica PLC-5 o SLC 500 alla logica Logix5550 – Manuale di riferimento*, pubblicazione 1756-6.8.5

A chi è rivolto questo manuale

Questo documento si rivolge ai programmatori per fornire loro dettagliate informazioni riguardo le istruzioni disponibili con i controllori Logix. È necessario essere già a conoscenza delle modalità di archiviazione ed elaborazione dati dei controllori Logix.

Si consiglia ai programmatori non esperti di leggere tutte le informazioni riguardanti un'istruzione prima di utilizzare la stessa. I programmatori esperti possono fare riferimento alle informazioni sulle istruzioni per verificare eventuali dettagli.

Scopo di questo manuale

Ogni descrizione è riportata nel seguente formato.

Questa sezione:	Fornisce questo tipo di informazioni:
Nome dell'istruzione	identifica l'istruzione specifica se si tratta di un'istruzione di ingresso o di uscita
Operandi	elenca tutti gli operandi dell'istruzione se disponibile in ladder, descrive gli operandi se disponibile in testo strutturato, descrive gli operandi se disponibile in blocco funzione, descrive gli operandi I pin indicati in un blocco funzione di default sono gli unici pin di default. Nella tabella degli operandi vengono elencati tutti i possibili pin relativi a un blocco funzione.
Struttura dell'istruzione	elenca i bit e i valori di stato del controllo dell'istruzione, se presenti
Descrizione	descrive l'uso dell'istruzione definisce le differenze tra quando un'istruzione è abilitata e quando è disabilitata
Indicatori di stato aritmetico	indica se l'istruzione influenza gli indicatori di stato aritmetico o meno vedere appendice Attributi comuni
Condizioni di errore	indica se l'istruzione genera errori gravi o minori in caso affermativo, viene indicato il tipo ed il codice di errore
Esecuzione	indica in che modo funziona l'istruzione
Esempio	fornisce almeno un esempio di programmazione comprende un descrizione esplicativa per ogni esempio

Le icone seguenti facilitano l'identificazione delle informazioni specifiche per i singoli linguaggi:

Questa icona:	Indica il seguente linguaggio di programmazione
	ladder
	testo strutturato
	blocco funzione

Informazioni comuni per tutte le istruzioni

Il set di istruzioni del Logix5000 presenta degli attributi comuni:

Per queste informazioni:	Vedere questa appendice:
attributi comuni	l'appendice Attributi comuni definisce: • indicatori di stato aritmetico • tipi di dati • parole chiave
attributi di un blocco funzione	l'appendice Attributi di un blocco funzione definisce: • istruzioni di controllo programma • modalità di temporizzazione

Convenzioni e termini correlati

Impostato e azzerato

In questo manuale si utilizzano i termini "impostato" ed "azzerato" per indicare lo stato dei bit (booleani) e dei valori (non booleani):

Questo termine:	Significa:
impostato	il bit è impostato a 1 (ON) un valore è impostato su un numero diverso da zero
azzerato	il bit è a 0 (OFF) tutti i bit di un valore sono a 0

Nella sezione relativa agli operandi, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimale, in genere DINT o REAL.

Condizione del ramo in logica ladder

Il controllore valuta le istruzioni ladder in base alla condizione del ramo precedente l'istruzione (condizione del ramo di ingresso). In base alla condizione di ingresso del ramo e all'istruzione, il controllore imposta la condizione del ramo seguente l'istruzione (condizione del ramo di uscita) che, a sua volta, determina la successiva istruzione.

Se la condizione di ingresso del ramo per un'istruzione di ingresso è vera, il controllore valuta l'istruzione ed imposta la condizione del ramo di uscita in base ai risultati dell'istruzione. Se l'istruzione è vera, la condizione del ramo di uscita è vera; se l'istruzione è falsa la condizione del ramo di uscita è falsa.

Il controllore esegue anche una prescansione delle istruzioni. La prescansione è una speciale scansione di tutte le routine di un controllore. Durante la prescansione, il controllore esegue una scansione di tutte le routine e subroutine, ma ignora i salti che potrebbero far sì che alcune istruzioni non vengano eseguite. Il controllore esegue tutti gli anelli FOR e le chiamate alle subroutine. Se una subroutine viene chiamata più di una volta, viene rieseguita ad ogni chiamata. Il controllore utilizza la prescansione delle istruzioni ladder per ripristinare valori di I/O non ritentivi e interni.

Durante la prescansione, i valori di ingresso non vengono applicati e i risultati non vengono scritti. La prescansione può essere generata nelle seguenti condizioni:

- Passaggio da modalità Programmazione a modalità di Esecuzione
- Attivazione automatica della modalità di esecuzione da una condizione di accensione.

La prescansione non viene eseguita quando:

- Il programma viene schedulato mentre il controllore è in esecuzione.
- Il programma viene cancellato mentre il controllore è in esecuzione.

Stati dei blocchi funzione

IMPORTANTE Quando si programma con blocchi funzione, limitare l'intervallo di unità ingegneristiche a $+/-10^{+/-15}$, dato che i calcoli interni a virgola mobile vengono eseguiti usando la virgola mobile a singola precisione. Le unità ingegneristiche che non rientrano in questo intervallo possono compromettere la precisione, se i risultati si avvicinano ai limiti della virgola mobile a singola precisione ($+/-10^{+/-38}$).

Il controllore valuta le istruzioni in blocco funzione sulla base dello stato delle varie condizioni.

Condizione possibile:	Descrizione
durante la prescansione	La prescansione nel caso delle routine in blocco funzione avviene con le stesse modalità delle routine in ladder. L'unica differenza sta nel fatto che il parametro EnableIn relativo a ciascuna istruzione di blocco funzione viene azzerato durante la prescansione.
prima scansione dell'istruzione	Questa dicitura si riferisce alla prima volta in cui l'istruzione viene eseguita dopo la prescansione. Il controllore la usa per leggere gli ingressi e stabilire in quale stato devono essere impostati.
durante la prima esecuzione dell'istruzione	Questa dicitura si riferisce alla prima volta in cui l'istruzione viene eseguita durante una nuova istanza di una struttura dati. Il controllore la utilizza per generare dei coefficienti e altri dati memorizzati che non vengono modificati relativamente a un blocco funzione in seguito al download iniziale.

Ogni istruzione di blocco funzione comprende anche i parametri EnableIn e EnableOut:

- le istruzioni di blocco funzione vengono eseguite normalmente quando viene impostato EnableIn.
- quando EnableIn viene azzerato, l'istruzione di blocco funzione esegue la logica prescansione, logica post-scansione, oppure salta l'esecuzione normale dell'algoritmo.
- EnableOut corrisponde a EnableIn, tuttavia, se durante l'esecuzione del blocco funzione viene rilevata una condizione di overflow anche EnableOut è azzerato.
- ogni volta che EnableIn passa da azzerato a impostato e viceversa l'esecuzione del blocco funzione viene ripresa dal punto in cui era stata interrotta. Tuttavia, esistono delle istruzioni di blocco funzione che determinano funzionalità speciali, come la ripetizione della inizializzazione, che si verifica quando EnableIn passa da azzerato a impostato. Per quanto riguarda le istruzioni di blocco funzione con parametri a base tempo, ogni volta che si imposta la modalità di temporizzazione Oversample, l'esecuzione dell'istruzione riprende sempre dal punto in cui è stata interrotta quando EnableIn passa da azzerato a impostato.

Se il parametro EnableIn non è cablato, l'istruzione viene sempre eseguita in modo normale e EnableIn rimane impostato. Se si azzera EnableIn, quest'ultimo verrà impostato all'esecuzione successiva dell'istruzione.

Notizen:

Istruzioni di allarmi ed eventi FactoryTalk basati su Logix
(ALMD, ALMA)

Introduzione

Queste istruzioni di allarme basate su Logix sono disponibili in ladder, testo strutturato e diagramma a blocchi funzione. Quando vengono utilizzate con il software FactoryTalk View SE, versione 5.0 e successive, tali istruzioni creano un sistema di allarme insieme al pacchetto di visualizzazione posseduto. Il controllore rileva le condizioni di allarme e registra gli eventi nei server di allarmi ed eventi di FactoryTalk View, che inviano gli allarmi ai client di Factory Talk View SE registrati per ricevere notifiche.

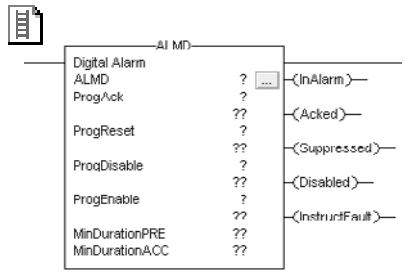
Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
rilevare allarmi in base a condizioni booleane (vero/falso)	ALMD	ladder testo strutturato blocco funzione	30
rilevare allarmi in base al livello o al tasso di variazione di un allarme	ALMA	ladder testo strutturato blocco funzione	42

Digital Alarm (ALMD)

L'istruzione ALMD rileva gli allarmi in base a condizioni booleane (vero/falso). I parametri di controllo di programma (Prog) e operatore (Oper) forniscono un'interfaccia per i comandi di allarme.

Operandi:

Ladder



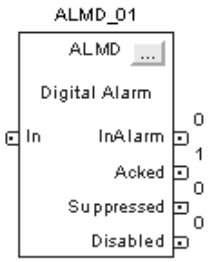
In logica ladder, l'ingresso della condizione di allarme (In) si ottiene dalla condizione del ramo.

Operando	Tipo	Formato	Descrizione
Tag ALMD	ALARM_DIGITAL	Struttura	Struttura ALMD.
In	BOOL	Tag immediato	**Solo testo strutturato.** Il valore viene copiato in In quando l'istruzione viene eseguita. Il valore di ingresso dell'allarme viene confrontato per determinare se esiste un allarme.
ProgAck	BOOL	Tag immediato	Il valore viene copiato in ProgAck quando l'istruzione viene eseguita. Nella transizione da azzerato a impostato, viene confermata la ricezione dell'allarme (se la conferma è richiesta).
ProgReset	BOOL	Tag immediato	Il valore viene copiato in ProgReset quando l'istruzione viene eseguita. Nella transizione da azzerato a impostato, viene azzerato l'allarme (se richiesto).
ProgDisable	BOOL	Tag immediato	Il valore viene copiato in ProgDisable quando l'istruzione viene eseguita. Quando è impostato, disattiva l'allarme (non annulla il comando di attivazione comandi).
ProgEnable	BOOL	Tag immediato	Il valore viene copiato in ProgEnable quando l'istruzione viene eseguita. Quando è impostato, attiva l'allarme (ha la precedenza sul comando di disattivazione comandi).
MinDurationPRE	DINT	immediato	**Solo logica ladder.** Specifica per quanto tempo la condizione di allarme deve essere presente prima che sia segnalata (millisecondi).
MinDurationACC	DINT	immediato	**Solo logica ladder.** Indica il numero di millisecondi che sono trascorsi da quando si è verificata la condizione di allarme.

Testo strutturato

```
ALMD(ALMD, In, ProgAck,
ProgReset, ProgDisable,
ProgEnable);
```

Gli operandi sono gli stessi dell'istruzione di logica ladder ALMD, con alcune eccezioni, come sopra indicato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
Tag ALMD	ALARM_DIGITAL	Struttura	Struttura ALMD.

Struttura ALARM_DIGITAL

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Logica ladder:** Corrisponde allo stato del ramo. Non incide sull'elaborazione. **Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato:** Nessun effetto. L'istruzione viene sempre eseguita.
In	BOOL	Ingresso del segnale digitale nell'istruzione. Il valore di default è azzerato. **Logica ladder:** Segue la condizione del ramo. Impostato se la condizione del ramo è vera. Azzerato se la condizione del ramo è falsa. **Testo strutturato:** Copiato dall'operando dell'istruzione.
InFault	BOOL	Indicatore di stato generale danneggiato per l'ingresso. L'applicazione utente può impostatare InFault per indicare che il segnale di ingresso presenta un errore. Quando è impostato, l'istruzione passa a InFaulted (Status.1). Quando è azzerato, l'istruzione azzera InFaulted (Status.1). In entrambi i casi, l'istruzione continua a valutare In per le condizioni di allarme. Il valore di default è azzerato (stato generale corretto).
Condition	BOOL	Specifica in che modo l'allarme viene attivato. Quando Condition è impostato, la condizione di allarme viene attivata quando In è impostato. Quando Condition è azzerato, la condizione di allarme viene attivata quando In è azzerato. Il valore di default è impostato.

Parametro di ingresso	Tipo di dati	Descrizione
AckRequired	BOOL	Specifica se è richiesta la conferma di ricezione dell'allarme. Se è impostato, è richiesta la conferma di ricezione dell'allarme. Se è azzerato, la conferma di ricezione dell'allarme non è richiesta e Acked è sempre impostato. Il valore di default è impostato.
Latched	BOOL	Specifica se l'allarme è agganciato. Gli allarmi agganciati restano nello stato InAlarm quando la condizione di allarme diventa falsa, finché non viene ricevuto un comando Reset di azzeramento. Quando è impostato, l'allarme è agganciato. Quando è azzerato, l'allarme è sganciato. Un allarme agganciato può essere azzerato solo quando la condizione di allarme è falsa. Il valore di default è azzerato.
ProgAck	BOOL	Impostato dal programma utente per confermare la ricezione dell'allarme. Richiede una transizione da azzerato a impostato mentre l'allarme non ha ancora ricevuto conferma di ricevimento. Il valore di default è azzerato. **Logica ladder:** Copiato dall'operando dell'istruzione. **Testo strutturato:** Copiato dall'operando dell'istruzione.
OperAck	BOOL	Impostato dall'interfaccia operatore per confermare la ricezione dell'allarme. Richiede una transizione da azzerato a impostato mentre l'allarme non ha ancora ricevuto conferma di ricevimento. L'istruzione azzera questo parametro. Il valore di default è azzerato.
ProgReset	BOOL	Impostato dal programma utente per azzerare l'allarme. Richiede una transizione da azzerato a impostato mentre l'allarme è in stato InAlarm e la condizione In non è in allarme. Il valore di default è azzerato. **Logica ladder:** Copiato dall'operando dell'istruzione. **Testo strutturato:** Copiato dall'operando dell'istruzione.
OperReset	BOOL	Impostato dall'interfaccia operatore per azzerare l'allarme. Richiede una transizione da azzerato a impostato mentre l'allarme è in stato InAlarm e la condizione In non è in allarme. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
ProgSuppress	BOOL	Impostato dal programma utente per sopprimere l'allarme. Il valore di default è azzerato.
OperSuppress	BOOL	Impostato dall'interfaccia operatore per sopprimere l'allarme. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.

Parametro di ingresso	Tipo di dati	Descrizione
ProgUnsuppress	BOOL	Impostato dal programma utente per annullare la soppressione dell'allarme. Ha la precedenza sui comandi di soppressione. Il valore di default è azzerato.
OperUnsuppress	BOOL	Impostato dall'interfaccia operatore per annullare la soppressione dell'allarme. Ha la precedenza sui comandi di soppressione. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
ProgDisable	BOOL	Impostato dal programma utente per disattivare l'allarme. Il valore di default è azzerato. **Logica ladder:** Copiato dall'operando dell'istruzione. **Testo strutturato:** Copiato dall'operando dell'istruzione.
OperDisable	BOOL	Impostato dall'interfaccia operatore per disabilitare l'allarme. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
ProgEnable	BOOL	Impostato dal programma utente per abilitare l'allarme. Ha la precedenza sul comando di disabilitazione. Il valore di default è azzerato. **Logica ladder:** Copiato dall'operando dell'istruzione. **Testo strutturato:** Copiato dall'operando dell'istruzione.
OperEnable	BOOL	Impostato dall'interfaccia operatore per abilitare l'allarme. Ha la precedenza sul comando di disabilitazione. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
AlarmCountReset	BOOL	Impostato dal programma utente per azzerare il conteggio dell'allarme. Una transizione da azzerato a impostato azzera il conteggio dell'allarme. Il valore di default è azzerato.
UseProgTime	BOOL	Specifica se utilizzare l'orologio del controllore o il valore ProgTime per la registrazione cronologica degli eventi di cambiamento dello stato dell'allarme. Quando è impostato, il valore ProgTime esegue la registrazione cronologica. Quando è azzerato, l'orologio del controllore esegue la registrazione cronologica. Il valore di default è azzerato.

Parametro di ingresso	Tipo di dati	Descrizione
ProgTime	LINT	Se UseProgTime è impostato, questo valore è utilizzato per fornire il valore di registrazione cronologica per tutti gli eventi. Questo consente all'applicazione di applicare le registrazioni cronologiche ottenute dalla sorgente dell'allarme, ad esempio un modulo d'ingresso di una sequenza di eventi.
Severity	DINT	Gravità dell'allarme. Questo non influenza l'elaborazione degli allarmi da parte del controllore, ma può essere utilizzato per ordinare e filtrare le funzioni di chi riceve l'allarme. Valori validi = 1...1000 (1000 = più grave; 1 = meno grave). Valore predefinito = 500.
MinDurationPRE	DINT	Valore preimpostato della durata minima (in millisecondi) per cui la condizione di allarme resta vera prima che l'allarme sia contrassegnato come InAlarm e una notifica di allarme sia inviata ai client. Il controllore raccoglie i dati dell'allarme non appena la condizione di allarme viene rilevata, pertanto non vi è alcuna perdita di dati mentre si attende la durata minima. Valori validi = 0...2.147.483.647. Valore predefinito = 0.

Parametro di uscita	Tipo di dati	Descrizione
EnableOut	BOOL	Abilita l'uscita.
InAlarm	BOOL	Stato di allarme attivo. Impostato quando l'allarme è attivo. Azzerato quando l'allarme non è attivo (stato normale).
Acked	BOOL	Stato di conferma di ricezione dell'allarme. Impostato quando è stata data conferma di ricezione dell'allarme. Azzerato quando non è stata data conferma di ricezione dell'allarme. Acked è sempre impostato quando AckRequired è azzerato.
InAlarmUnack	BOOL	Combinazione di allarme attivo e stato di conferma ricezione. Impostato quando l'allarme è attivo (InAlarm è impostato) e non è stata data conferma di ricevimento dell'allarme (Acked è azzerato). Azzerato quando l'allarme è normale (inattivo), è stata data conferma di ricezione dell'allarme o entrambi gli stati sono presenti.
Suppressed	BOOL	Stato di allarme soppresso. Impostato quando l'allarme è soppresso. Azzerato quando l'allarme non è soppresso.
Disabled	BOOL	Stato di allarme disabilitato. Impostato quando l'allarme è disabilitato. Azzerato quando l'allarme è abilitato.
MinDurationACC	DINT	Tempo trascorso dal rilevamento dell'allarme. Quando il valore raggiunge MinDurationPRE, l'allarme diventa attivo (InAlarm è impostato) e una notifica viene inviata ai client.
AlarmCount	DINT	Numero di attivazioni dell'allarme (InAlarm impostato). Se viene raggiunto il valore massimo, il contatore lascia il valore al valore di conteggio massimo.
InAlarmTime	LINT	Registrazione cronologica del rilevamento dell'allarme.
AckTime	LINT	Registrazione cronologica della conferma dell'allarme. Se l'allarme non richiede conferma, questa registrazione cronologica è uguale all'ora dell'allarme.
RetToNormalTime	LINT	Registrazione cronologica del ritorno dell'allarme allo stato normale.
AlarmCountResetTime	LINT	Registrazione cronologica che indica quando è stato azzerato il conteggio dell'allarme.
DeliveryER	BOOL	Errore di consegna del messaggio di notifica dell'allarme. Impostato in caso di errore di consegna: non è registrato alcun ricevente degli allarmi o almeno un ricevente registrato non ha ricevuto l'ultimo messaggio di modifica dello stato dell'allarme. Azzerato quando la consegna è avvenuta con successo o è in corso.

Parametro di uscita	Tipo di dati	Descrizione
DeliveryDN	BOOL	Completamento della consegna del messaggio di notifica dell'allarme. Impostato quando la consegna riesce correttamente: esiste almeno un ricevente registrato e tutti i riceventi registrati hanno ricevuto correttamente l'ultimo messaggio di cambiamento di stato dell'allarme. Azzerato quando la consegna non viene portata a termine correttamente o è in corso.
DeliveryEN	BOOL	Stato di consegna del messaggio di notifica dell'allarme. Impostato quando la consegna è in corso. Azzerato quando la consegna non è in corso.
NoSubscriber	BOOL	L'allarme non aveva riceventi registrati quando ha tentato di consegnare il messaggio più recente. Impostato quando non vi sono riceventi registrati. Azzerato quando esiste almeno un ricevente registrato.
NoConnection	BOOL	I riceventi registrati dell'allarme non erano collegati durante il tentativo di consegnare il messaggio più recente. Impostato quando tutti i riceventi registrati sono scollegati. Azzerato quando almeno un ricevente è collegato o non vi sono riceventi registrati.
CommError	BOOL	Errore di comunicazione nella consegna di un messaggio di allarme. Impostato quando vi sono errori di comunicazione e vengono utilizzati tutti i tentativi. Questo significa che esiste un ricevente registrato con una connessione, ma il controllore non ha ricevuto la conferma della consegna del messaggio. Azzerato quando tutti i riceventi registrati collegati confermano la ricezione del messaggio di allarme.
AlarmBuffered	BOOL	Messaggio di allarme bufferizzato a causa di un errore di comunicazione (CommError è impostato) o connessione interrotta (NoConnection è impostato). Impostato quando il messaggio di allarme è bufferizzato per almeno un ricevente registrato. Azzerato quando il messaggio di allarme non è bufferizzato.
Subscribers	DINT	Numero di riceventi registrati per l'allarme.
SubscNotified	DINT	Numero di riceventi registrati notificati correttamente del cambiamento di stato dell'allarme più recente.
Status	DINT	Indicatori di stato combinati: Status.0 = InstructFault. Status.1 = InFaulted. Status.2 = SeverityInv.
InstructFault (Status.0)	BOOL	Esistono condizioni di errore dell'istruzione. Non si tratta di un errore minore o grave del controllore. Verificare gli altri bit di stato per stabilire cosa è accaduto.
InFaulted (Status.1)	BOOL	Il programma utente ha impostato InFault per indicare dati in ingresso di cattiva qualità. L'allarme continua a valutare In per la condizione di allarme.
SeverityInv (Status.2)	BOOL	La configurazione della gravità dell'allarme non è valida. Se la gravità è <1, l'istruzione usa Severity = 1. Se la gravità è >1000, l'istruzione usa Severity = 1000.

Descrizione

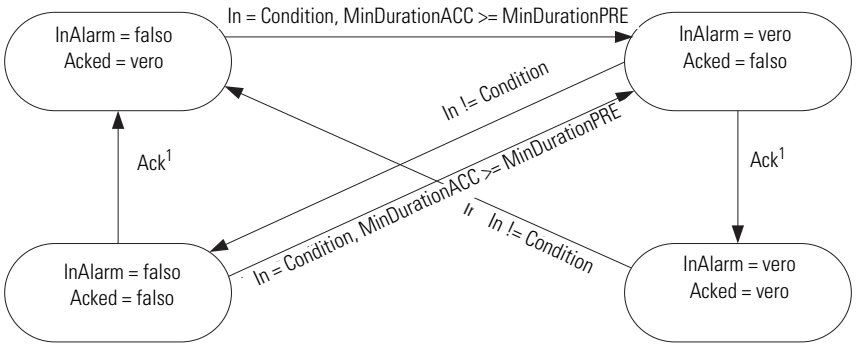
L'istruzione ALMD rileva gli allarmi in base a condizioni booleane (vero/falso).

L'istruzione ALMD offre funzioni aggiuntive quando è utilizzata con i software RSLinx Enterprise e FactoryTalk View SE. È possibile visualizzare allarmi nelle visualizzazioni Alarm Summary, Alarm Banner, Alarm Status Explorer e Alarm Log Viewer del software FactoryTalk View SE.

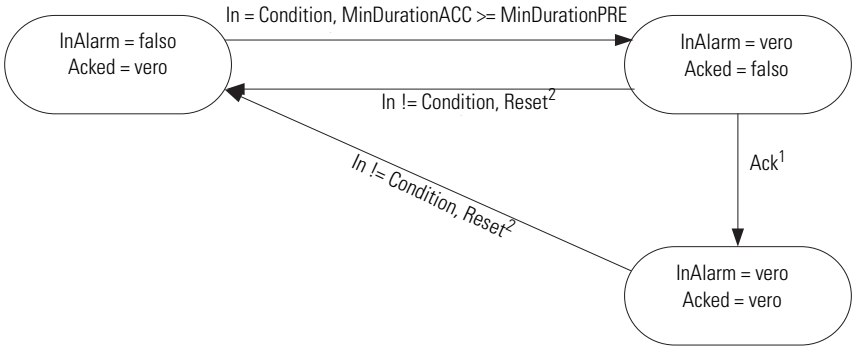
Il software RSLinx Enterprise è ricevente registrato degli allarmi nel controllore. Usare i parametri di uscita per monitorare l'istruzione, vedere lo stato dei riceventi registrati dell'allarme e visualizzare i cambiamenti di stato dell'allarme. Se la connessione al software RSLinx Enterprise si interrompe, il controllore può brevemente bufferizzare i dati di allarme finché la connessione viene ripristinata.

Diagrammi di stato quando è richiesta la conferma di ricezione

Latched = falso



Latched = vero

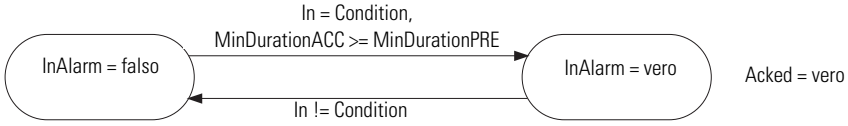


[1] La ricezione dell'allarme può essere confermata in diversi modi: ProgAck, OperAck, client (software RSLogix 5000, software RSView).

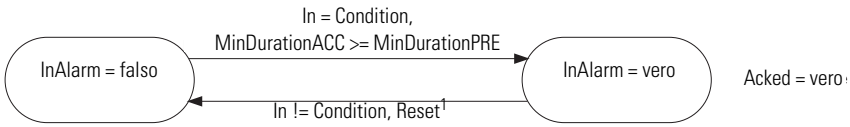
[2] L'allarme può essere azzerato in diversi modi: ProgReset, OperReset, client (software RSLogix 5000, software RSView).

Diagrammi di stato quando la conferma della ricezione non è richiesta

Latched = falso



Latched = vero



[1] L'allarme può essere azzerato in diversi modi: ProgReset, OperReset, client (software RSLogix 5000, software RSView)

Indicatori di stato aritmetico: nessuno

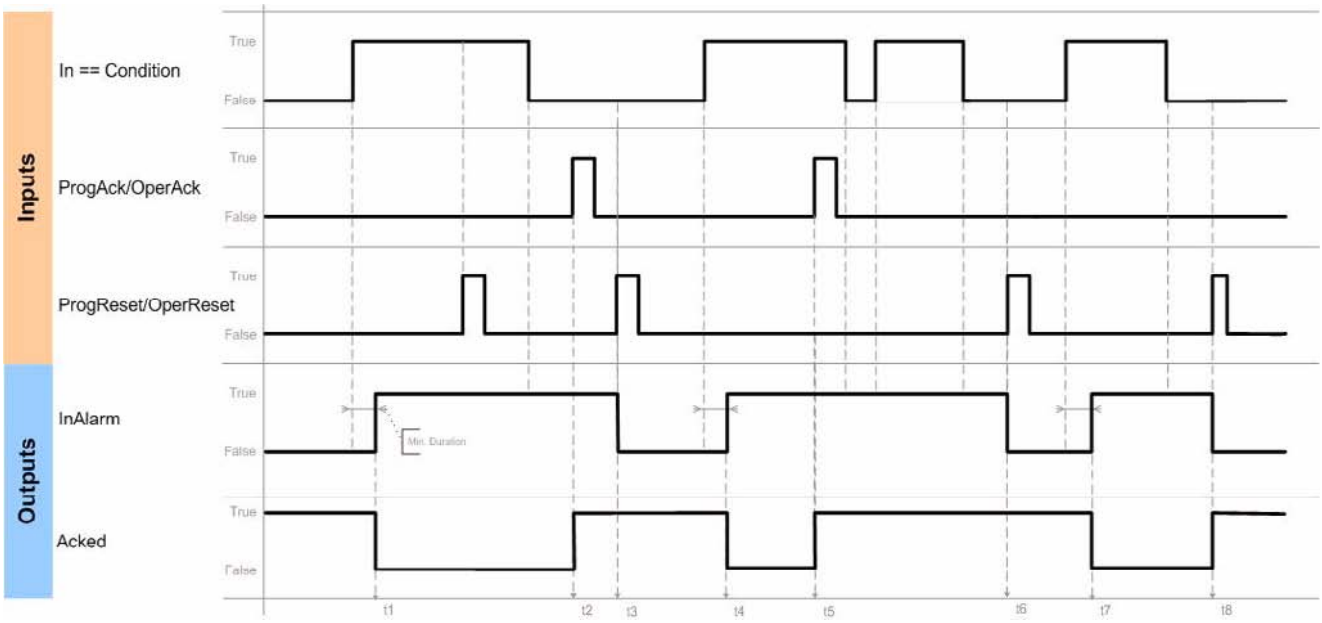
Condizioni di errore: nessuna

Esecuzione:

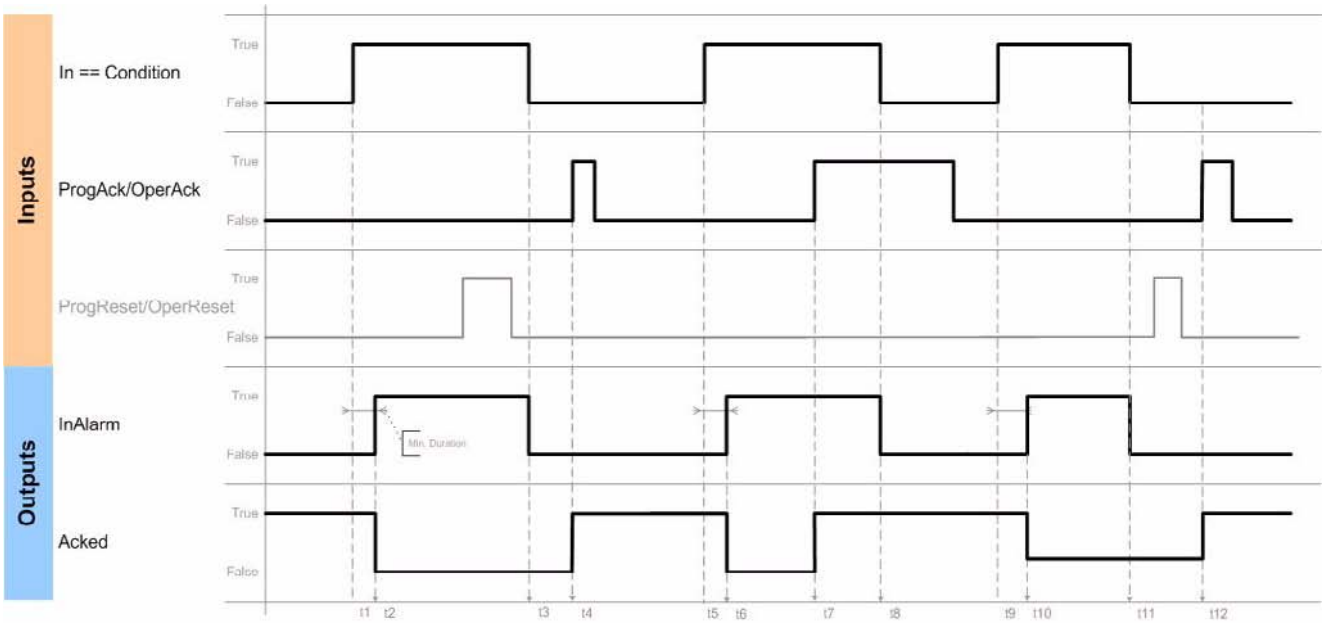
Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso. InAlarm viene azzerato and Acked viene impostato. Tutte le richieste operatore, le registrazioni cronologiche e gli indicatori di consegna vengono azzerati.
la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso. EnableIn e EnableOut vengono azzerati. Il parametro In viene azzerato e l'istruzione esegue una valutazione per determinare lo stato dell'allarme.
la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su vero. EnableIn e EnableOut vengono impostati. Il parametro In viene impostato e l'istruzione esegue una valutazione per determinare lo stato dell'allarme.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Tutte le richieste operatore, le registrazioni cronologiche e gli indicatori di consegna vengono azzerati. InAlarm viene azzerato and Acked viene impostato.	Tutte le richieste operatore, le registrazioni cronologiche e gli indicatori di consegna vengono azzerati. InAlarm viene azzerato and Acked viene impostato.
prima scansione dell'istruzione	Nessuna.	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.	Nessuna.
EnableIn viene azzerato	L'istruzione non viene eseguita. EnableOut viene azzerato.	L'istruzione viene eseguita. EnableOut viene sempre impostato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.	L'istruzione viene eseguita. EnableOut viene sempre impostato.
post-scansione	Nessuna.	Nessuna.

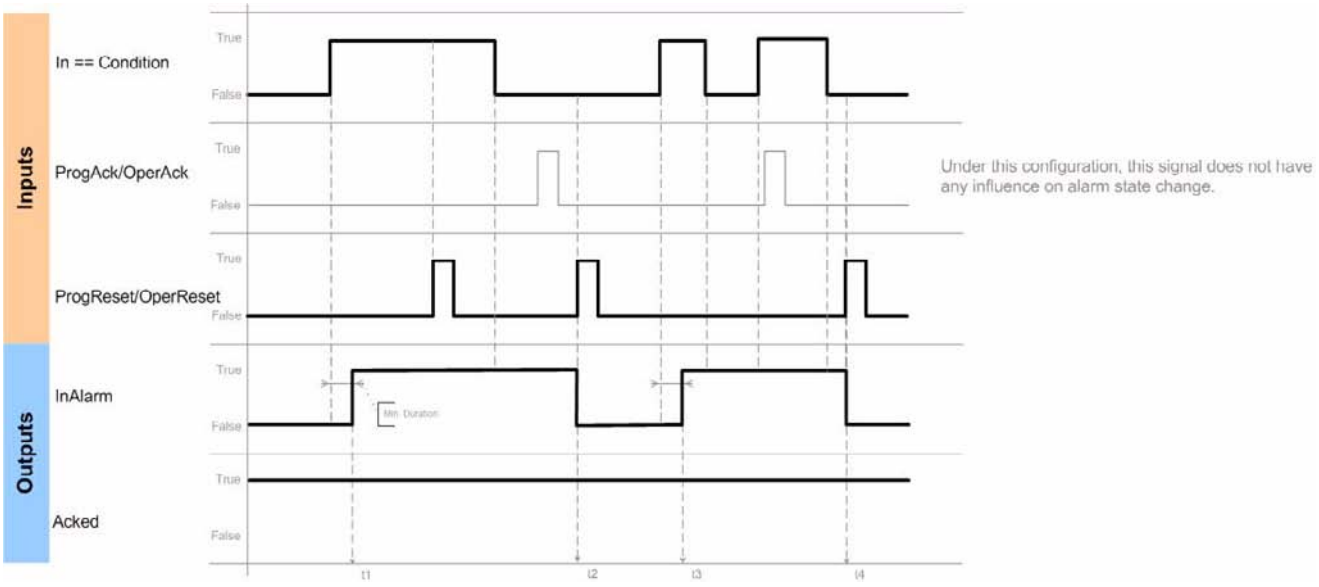
Allarme ALMD, conferma di ricezione richiesta e agganciato



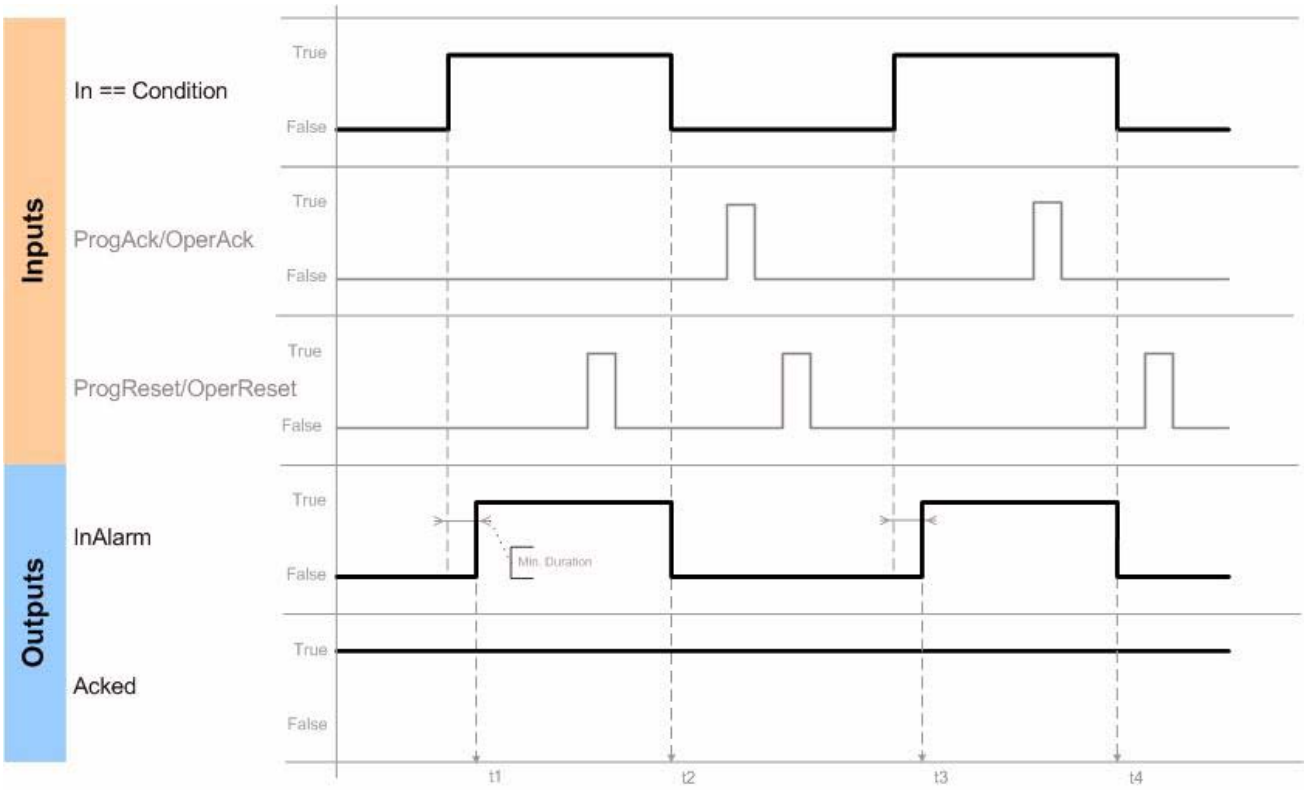
Allarme ALMD, conferma di ricezione richiesta e non agganciato



Allarme ALMD, conferma di ricezione non richiesta e agganciato

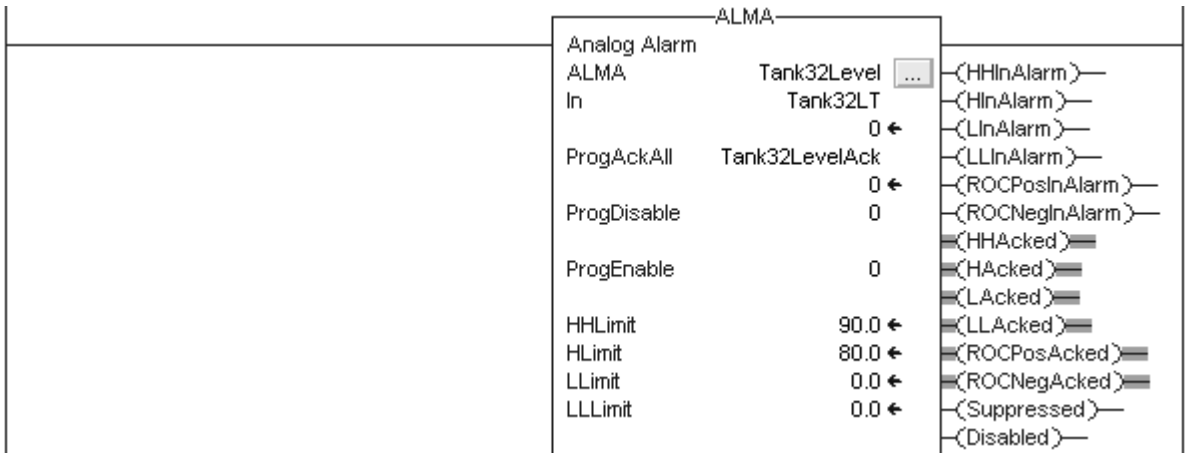


Allarme ALMD, conferma di ricezione non richiesta e non agganciato



Esempio: due segnali di errore del motore sono combinati in modo tale che, nel caso in cui uno dei due si verifichi, viene attivato un allarme motore. Confermare via programma la ricezione di tutte le condizioni di allarme con una transizione da azzerato a impostato del valore del tag Motor101Ack. La logica applicativa deve azzerare Motor101Ack.

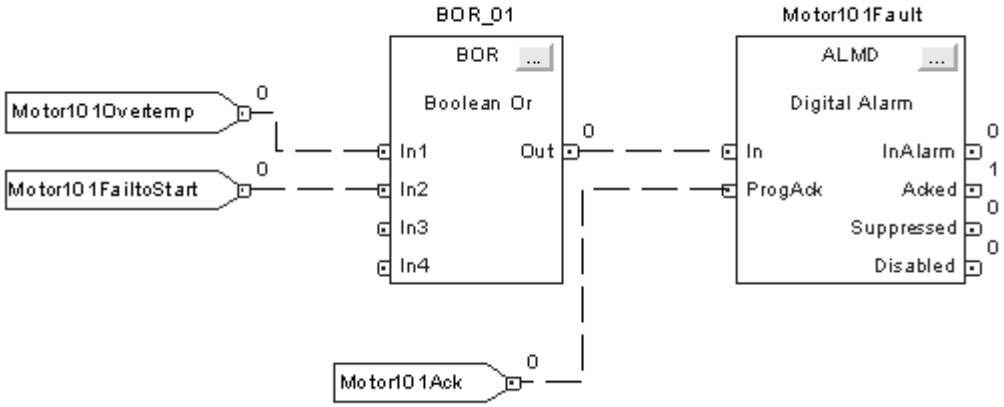
Logica ladder



Testo strutturato

```
Motor101FaultConditions := Motor101Overtemp OR
Motor101FailToStart;

ALMD(Motor101Fault,Motor101FaultConditions,Motor101Ack,
0,0,0 );
```

Blocco funzione

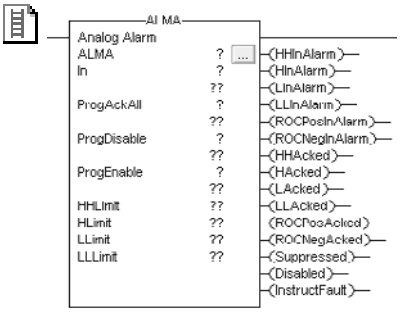


Analog Alarm (ALMA)

L'istruzione ALMA rileva gli allarmi in base al livello o al tasso di variazione di un valore analogico. I parametri di controllo di programma (Prog) e operatore (Oper) forniscono un'interfaccia per i comandi di allarme.

Operandi:

Ladder



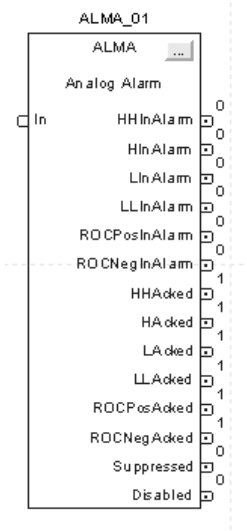
Operando	Tipo	Formato	Descrizione
tag ALMA	ALARM_ANALOG	Struttura	Struttura ALMA.
In	REAL DINT INT SINT	Tag immediato	Il valore viene copiato in In quando l'istruzione viene eseguita. Valore dell'ingresso dell'allarme, confrontato con i limiti di allarme per rilevare le condizioni di allarme.
ProgAckAll	BOOL	Tag immediato	Il valore viene copiato in ProgAckAll quando l'istruzione viene eseguita. Nella transizione da azzerato a impostato, conferma la ricezione di tutte le condizioni di allarme che richiedono conferma.
ProgDisable	BOOL	Tag immediato	Il valore viene copiato in ProgDisable quando l'istruzione viene eseguita. Quando è impostato, disattiva l'allarme (non annulla il comando di attivazione comandi).
ProgEnable	BOOL	Tag immediato	Il valore viene copiato in ProgEnable quando l'istruzione viene eseguita. Quando è impostato, attiva l'allarme (ha la precedenza sul comando di disattivazione comandi).
HHlimit	REAL	immediato	**Solo logica ladder.** Limite di allarme massimo.
HLimit	REAL	immediato	**Solo logica ladder.** Limite di allarme alto.
LLimit	REAL	immediato	**Solo logica ladder.** Limite di allarme basso.
LLLimit	REAL	immediato	**Solo logica ladder.** Limite di allarme minimo.

Testo strutturato

```
ALMA(ALMA, In, ProgAckAll,
ProgDisable, ProgEnable);
```

Gli operandi sono gli stessi dell'istruzione di logica ladder ALMD, con alcune eccezioni, come sopra indicato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
Tag ALMA	ALARM_ANALOG	Struttura	Struttura ALMA

Struttura ALARM_ANALOG

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Logica ladder:** Corrisponde allo stato del ramo. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. **Testo strutturato:** Nessun effetto. L'istruzione viene sempre eseguita. **Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
In	REAL	Il valore dell'ingresso dell'allarme, confrontato con i limiti di allarme per rilevare le condizioni di allarme. Valore predefinito = 0.0. **Logica ladder:** Copiato dall'operando dell'istruzione. **Testo strutturato:** Copiato dall'operando dell'istruzione.
InFault	BOOL	Indicatore di stato danneggiato per l'ingresso. L'applicazione utente può impostare InFault per indicare che il segnale di ingresso presenta un errore. Quando è impostato, l'istruzione passa a InFaulted (stato.1). Quando è azzerato, l'istruzione azzera InFaulted (stato .1). In entrambi i casi, l'istruzione continua a valutare In per le condizioni di allarme. Il valore di default è azzerato (stato generale corretto).
HHEnabled	BOOL	Rilevamento condizione di allarme massimo. Impostare per abilitare il rilevamento della condizione di allarme massimo. Azzerare per disabilitare il rilevamento della condizione di allarme massimo. Il valore di default è impostato.
HEnabled	BOOL	Rilevamento condizione di allarme alto. Impostare per abilitare il rilevamento della condizione di allarme alto. Azzerare per disabilitare il rilevamento della condizione di allarme alto. Il valore di default è impostato.
LEnabled	BOOL	Rilevamento condizione di allarme basso. Impostare per abilitare il rilevamento della condizione di allarme basso. Azzerare per disabilitare il rilevamento della condizione di allarme basso. Il valore di default è impostato.
LLEnabled	BOOL	Rilevamento condizione di allarme minimo. Impostare per abilitare il rilevamento della condizione di allarme minimo. Azzerare per disabilitare il rilevamento della condizione di allarme minimo. Il valore di default è impostato.

Parametro di ingresso	Tipo di dati	Descrizione
AckRequired	BOOL	Specifica se è richiesta la conferma di ricezione dell'allarme. Se è impostato, è richiesta la conferma di ricezione dell'allarme. Quando viene azzerato, non è richiesta la conferma di ricezione e HHAcked, HAcked, LAcked, LLAcked, ROCPosAcked e ROCNegAcked sono sempre impostati. Il valore di default è impostato.
ProgAckAll	BOOL	Impostato dal programma utente per confermare tutte le condizioni dell'allarme. Richiede una transizione da azzerato a impostato mentre le condizioni dell'allarme non sono confermate. Il valore di default è azzerato. **Logica ladder:** Copiato dall'operando dell'istruzione. **Testo strutturato:** Copiato dall'operando dell'istruzione.
OperAckAll	BOOL	Impostato dall'interfaccia operatore per confermare tutte le condizioni dell'allarme. Richiede una transizione da azzerato a impostato mentre le condizioni dell'allarme non sono confermate. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
HHProgAck	BOOL	Conferma programma massimo. Impostato dal programma utente per confermare una condizione massima. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. Il valore di default è azzerato.
HHOperAck	BOOL	Conferma operatore massimo. Impostato dall'interfaccia operatore per confermare una condizione massima. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
HProgAck	BOOL	Conferma programma alto. Impostato dal programma utente per confermare una condizione alta. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. Il valore di default è azzerato.
HOperAck	BOOL	Conferma operatore alto. Impostato dall'interfaccia operatore per confermare una condizione alta. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
LProgAck	BOOL	Conferma programma basso. Impostato dal programma utente per confermare una condizione bassa. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. Il valore di default è azzerato.
LOperAck	BOOL	Conferma operatore basso. Impostato dall'interfaccia operatore per confermare una condizione bassa. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.

Parametro di ingresso	Tipo di dati	Descrizione
LLProgAck	BOOL	Conferma programma minimo. Impostato dal programma utente per confermare una condizione minima. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. Il valore di default è azzerato.
LLOperAck	BOOL	Conferma operatore minimo. Impostato dall'interfaccia operatore per confermare una condizione minima. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
ROCPosProgAck	BOOL	Conferma del programma di tasso di variazione positiva. Impostato dal programma utente per confermare una condizione di tasso di variazione positiva. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. Il valore di default è azzerato.
ROCPosOperAck	BOOL	Conferma dell'operatore di tasso di variazione positiva. Impostato dall'interfaccia operatore per confermare una condizione di tasso di variazione positiva. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
ROCNegProgAck	BOOL	Conferma del programma di tasso di variazione negativa. Impostato dal programma utente per confermare una condizione di tasso di variazione negativa. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. Il valore di default è azzerato.
ROCNegOperAck	BOOL	Conferma dell'operatore di tasso di variazione negativa. Impostato dall'interfaccia operatore per confermare una condizione di tasso di variazione negativa. Richiede una transizione da azzerato a impostato mentre la condizione di allarme non ha ancora ricevuto conferma di ricevimento. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
ProgSuppress	BOOL	Impostato dal programma utente per sopprimere l'allarme. Il valore di default è azzerato.
OperSuppress	BOOL	Impostato dall'interfaccia utente per sopprimere l'allarme. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
ProgUnsuppress	BOOL	Impostato dal programma utente per annullare la soppressione dell'allarme. Ha la precedenza sui comandi di soppressione. Il valore di default è azzerato.
OperUnsuppress	BOOL	Impostato dall'interfaccia utente per annullare la soppressione dell'allarme. Ha la precedenza sui comandi di soppressione. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.

Parametro di ingresso	Tipo di dati	Descrizione
ProgDisable	BOOL	Impostato dal programma utente per disattivare l'allarme. Il valore di default è azzerato. **Logica ladder:** Copiato dall'operando dell'istruzione. **Testo strutturato:** Copiato dall'operando dell'istruzione.
OperDisable	BOOL	Impostato dall'interfaccia operatore per disabilitare l'allarme. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
ProgEnable	BOOL	Impostato dal programma utente per abilitare l'allarme. Ha la precedenza sul comando di disabilitazione. Il valore di default è azzerato. **Logica ladder:** Copiato dall'operando dell'istruzione. **Testo strutturato:** Copiato dall'operando dell'istruzione.
OperEnable	BOOL	Impostato dall'interfaccia operatore per abilitare l'allarme. Ha la precedenza sul comando di disabilitazione. L'istruzione di allarme azzera questo parametro. Il valore di default è azzerato.
AlarmCountReset	BOOL	Impostato dal programma utente per azzerare i conteggi dell'allarme per tutte le condizioni. Una transizione da azzerato a impostato azzera i conteggi dell'allarme. Il valore di default è azzerato.
HHlimit	REAL	Limite di allarme massimo. Valori validi = HLimit < HHLimit < valore a virgola mobile positivo massimo. Valore predefinito = 0.0.
HHSeverity	DINT	Gravità della condizione di allarme massimo. Questo non influenza l'elaborazione degli allarmi da parte del controllore, ma può essere utilizzato per ordinare e filtrare le funzioni di chi riceve l'allarme. Valori validi = 1...1000 (1000 = più grave; 1 = meno grave). Valore predefinito = 500.
HLimit	REAL	Limite di allarme alto. Valori validi = LLimit < HLimit < HHLimit. Valore predefinito = 0.0.

Parametro di ingresso	Tipo di dati	Descrizione
HSeverity	DINT	Gravità della condizione di allarme alto. Questo non influenza l'elaborazione degli allarmi da parte del controllore, ma può essere utilizzato per ordinare e filtrare le funzioni di chi riceve l'allarme. Valori validi = 1...1000 (1000 = più grave; 1 = meno grave). Valore predefinito = 500.
LLimit	REAL	Limite di allarme basso. Valori validi = LLLimit < LLimit < HLimit. Valore predefinito = 0.0.
LSeverity	DINT	Gravità della condizione di allarme basso. Questo non influenza l'elaborazione degli allarmi da parte del controllore, ma può essere utilizzato per ordinare e filtrare le funzioni di chi riceve l'allarme. Valori validi = 1...1000 (1000 = più grave; 1 = meno grave). Valore predefinito = 500.
LLLimit	REAL	Limite di allarme minimo. Valori validi = valore a virgola mobile negativo massimo < LLLimit < LLimit. Valore predefinito = 0.0.
LLSeverity	DINT	Gravità della condizione di allarme minimo. Questo non influenza l'elaborazione degli allarmi da parte del controllore, ma può essere utilizzato per ordinare e filtrare le funzioni di chi riceve l'allarme. Valori validi = 1...1000 (1000 = più grave; 1 = meno grave). Valore predefinito = 500.
MinDurationPRE	DINT	Valore preimpostato di durata minima (millisecondi) per cui una condizione di livello di allarme deve rimanere vera prima che la condizione sia contrassegnata come InAlarm e sia inviata una notifica di allarme ai client. Il controllore raccoglie i dati dell'allarme non appena la condizione di allarme viene rilevata, pertanto non vi è alcuna perdita di dati mentre si attende la durata minima. Non si applica alle condizioni che riguardano il tasso di variazione. MinDurationPRE si applica solo al primo discostamento dal normale in entrambe le direzioni. Ad esempio, una volta che si verifica il timeout della condizione alta, la condizione massima diventa immediatamente attiva, mentre una condizione bassa aspetta il periodo di timeout. Valori validi = 0...2.147.483.647. Valore predefinito = 0.

Parametro di ingresso	Tipo di dati	Descrizione
Deadband	REAL	Banda morta per il rilevamento del ritorno allo stato normale dei livelli di allarme massimo, alto, basso e minimo. Una banda morta diversa da zero può ridurre il chattering della condizione di allarme se il valore In cambia continuamente ma rimane vicino alla soglia della condizione del livello. Il valore della banda morta non incide sulla transizione allo stato InAlarm (attivo). Una volta che una condizione di livello è attiva, prima che la condizione torni allo stato inattivo (normale), il valore In deve: • scende al di sotto della soglia meno la banda morta (per le condizioni alta e massima) o • sale al di sopra della soglia più la banda morta (per le condizioni bassa e minima). La banda morta non è utilizzata per condizionare la misura del tempo di durata minima. Valori validi = 0 ≤ banda morta < intervallo dal primo allarme basso abilitato al primo allarme alto abilitato. Valore predefinito = 0.0.
ROCPosLimit	REAL	Limite per un tasso di variazione crescente espresso in unità al secondo. Il rilevamento è abilitato per qualsiasi valore > 0.0 se ROCPeriod è anche > 0.0. Valori validi = 0.0...valore a virgola mobile massimo possibile. Valore predefinito = 0.0.
ROCPosSeverity	DINT	Gravità della condizione del tasso di variazione crescente. Questo non influenza l'elaborazione degli allarmi da parte del controllore, ma può essere utilizzato per ordinare e filtrare le funzioni di chi riceve l'allarme. Valori validi = 1...1000 (1000 = più grave; 1 = meno grave). Valore predefinito = 500.
ROCNegLimit	REAL	Limite per un tasso di variazione decrescente espresso in unità al secondo. Il rilevamento è abilitato per qualsiasi valore > 0.0 se ROCPeriod è anche > 0.0. Valori validi = 0.0...valore a virgola mobile massimo possibile. Valore predefinito = 0.0.
ROCNegSeverity	DINT	Gravità della condizione del tasso di variazione decrescente. Questo non influenza l'elaborazione degli allarmi da parte del controllore, ma può essere utilizzato per ordinare e filtrare le funzioni di chi riceve l'allarme. Valori validi = 1...1000 (1000 = più grave; 1 = meno grave). Valore predefinito = 500.
ROCPeriod	REAL	Periodo di tempo in secondi per il calcolo (intervallo di campionamento) del valore del tasso di variazione. Ogni volta che l'intervallo di campionamento termina, viene memorizzato un nuovo campione di In e ROC viene ricalcolato. Il rilevamento del tasso di variazione è abilitato per qualsiasi valore > 0.0. Valori validi = 0.0...valore a virgola mobile massimo possibile. Valore predefinito = 0.0.

Parametro di uscita	Tipo di dati	Descrizione
EnableOut	BOOL	Abilita l'uscita.
InAlarm	BOOL	Stato di allarme attivo. Impostato quando qualsiasi condizione di allarme è attiva. Azzerato quando tutte le condizioni di allarme non sono attive (stato normale).
AnyInAlarmUnack	BOOL	Combinazione di allarme attivo e stato di conferma ricezione. Impostato quando viene rilevata e confermata una qualsiasi condizione di allarme. Azzerato quando tutte le condizioni di allarme sono normali (inattive), confermate o entrambi gli stati sono presenti.
HHInAlarm	BOOL	Stato condizione di allarme massima. Impostato quando esiste una condizione massima. Azzerato quando non esiste una condizione massima.
HInAlarm	BOOL	Stato condizione di allarme alta. Impostato quando esiste una condizione alta. Azzerato quando non esiste una condizione alta.
LInAlarm	BOOL	Stato condizione di allarme bassa. Impostato quando esiste una condizione bassa. Azzerato quando non esiste una condizione bassa.
LLInAlarm	BOOL	Stato condizione di allarme minima. Impostato quando esiste una condizione minima. Azzerato quando non esiste una condizione minima.
ROCPosInAlarm	BOOL	Stato condizione di allarme tasso di variazione positiva. Impostato quando esiste una condizione di tasso di variazione positiva. Azzerato quando non esiste una condizione di tasso di variazione positiva.
ROCNegInAlarm	BOOL	Stato condizione di allarme tasso di variazione negativa. Impostato quando esiste una condizione di tasso di variazione negativa. Azzerato quando non esiste una condizione di tasso di variazione negativa.
ROC	REAL	Tasso di variazione calcolato del valore In. Questo valore viene aggiornato quando l'istruzione è sottoposta a scansione dopo ogni ROCPeriod trascorso. Il valore ROC è utilizzato per valutare le condizioni ROCPosInAlarm e ROCNegInAlarm. ROC = (campione corrente di In – campione precedente di In)/ROCPeriod
HHAcked	BOOL	Stato confermato condizione massima. Impostato quando una condizione massima è confermata. Sempre impostato quando AckRequired è azzerato. Azzerato quando una condizione massima non è confermata.
HAcked	BOOL	Stato condizione alta confermata. Impostato quando una condizione alta è confermata. Sempre impostato quando AckRequired è azzerato. Azzerato quando una condizione alta non è confermata.
LAcked	BOOL	Stato condizione bassa confermata. Impostato quando una condizione bassa è confermata. Sempre impostato quando AckRequired è azzerato. Azzerato quando una condizione bassa non è confermata.
LLAcked	BOOL	Stato condizione minima confermata. Impostato quando una condizione minima è confermata. Sempre impostato quando AckRequired è azzerato. Azzerato quando una condizione minima non è confermata.
ROCPosAcked	BOOL	Stato condizione tasso di variazione positiva confermata. Impostato quando una condizione di tasso di variazione positiva è confermata. Sempre impostato quando AckRequired è azzerato. Azzerato quando una condizione di tasso di variazione positiva non è confermata.
ROCNegAcked	BOOL	Stato condizione tasso di variazione negativa confermata. Impostato quando una condizione di tasso di variazione negativa è confermata. Sempre impostato quando AckRequired è azzerato. Azzerato quando una condizione di tasso di variazione negativa non è confermata.
HHInAlarmUnack	BOOL	Combinazione di condizione massima attiva e non confermata. Impostato quando la condizione massima è attiva (HHInAlarm è impostato) e non confermata. Azzerato quando la condizione massima è normale (inattiva), confermata o entrambe le cose.
HInAlarmUnack	BOOL	Combinazione di condizione alta attiva e non confermata. Impostato quando la condizione alta è attiva (HInAlarm è impostato) e non confermata. Azzerato quando la condizione alta è normale (inattiva), confermata o entrambe le cose.

Parametro di uscita	Tipo di dati	Descrizione
LInAlarmUnack	BOOL	Combinazione di condizione bassa attiva e non confermata. Impostato quando la condizione bassa è attiva (LInAlarm è impostato) e non confermata. Azzerato quando la condizione bassa è normale (inattiva), confermata o entrambe le cose.
LLInAlarmUnack	BOOL	Combinazione di condizione minima attiva e non confermata. Impostato quando la condizione minima è attiva (LLInAlarm è impostato) e non confermata. Azzerato quando la condizione minima è normale (inattiva), confermata o entrambe le cose.
ROCPosInAlarmUnack	BOOL	Combinazione di condizione di tasso di variazione positiva attiva e non confermata. Impostato quando la condizione di tasso di variazione positiva è attiva (ROCPosInAlarm è impostato) e non confermata. Azzerato quando la condizione di tasso di variazione positiva è normale (inattiva), confermata o entrambe le cose.
ROCNegInAlarmUnack	BOOL	Combinazione di condizione di tasso di variazione negativa attiva e non confermata. Impostato quando la condizione di tasso di variazione negativa è attiva (ROCNegInAlarm è impostato) e non confermata. Azzerato quando la condizione di tasso di variazione negativa è normale (inattiva), confermata o entrambe le cose.
Suppressed	BOOL	Stato di allarme soppresso. Impostato quando l'allarme è soppresso. Azzerato quando l'allarme non è soppresso.
Disabled	BOOL	Stato di allarme disabilitato. Impostato quando l'allarme è disabilitato. Azzerato quando l'allarme è abilitato.
MinDurationACC	DINT	Tempo trascorso dal rilevamento di una condizione di allarme. Quando questo valore raggiunge MinDurationPRE, tutte le condizioni di livello allarme rilevate diventano attive (*x*InAlarm è impostato) e una notifica viene inviata ai client.
HHInAlarmTime	LINT	Registrazione cronologica del momento in cui l'istruzione ALMA ha rilevato che il valore In ha superato il limite della condizione massima per la transizione più recente allo stato attivo.
HHAlarmCount	DINT	Il numero di volte in cui la condizione massima è stata attivata. Se viene raggiunto il valore massimo, il contatore lascia il valore al valore di conteggio massimo.
HInAlarmTime	LINT	Registrazione cronologica del momento in cui l'istruzione ALMA ha rilevato che il valore In ha superato il limite della condizione alta per la transizione più recente allo stato attivo.
HAlarmCount	DINT	Il numero di volte in cui la condizione alta è stata attivata. Se viene raggiunto il valore massimo, il contatore lascia il valore al valore di conteggio massimo.
LInAlarmTime	LINT	Registrazione cronologica del momento in cui l'istruzione ALMA ha rilevato che il valore In ha superato il limite della condizione bassa per la transizione più recente allo stato attivo.
LAlarmCount	DINT	Il numero di volte in cui la condizione bassa è stata attivata. Se viene raggiunto il valore massimo, il contatore lascia il valore al valore di conteggio massimo.
LLInAlarmTime	LINT	Registrazione cronologica del momento in cui l'istruzione ALMA ha rilevato che il valore In ha superato il limite della condizione minima per la transizione più recente allo stato attivo.
LLAlarmCount	DINT	Il numero di volte in cui la condizione minima è stata attivata. Se viene raggiunto il valore massimo, il contatore lascia il valore al valore di conteggio massimo.
ROCPosInAlarmTime	LINT	Registrazione cronologica de momento in cui l'istruzione ALMA ha rilevato che il valore In ha superato il limite della condizione d tasso di variazione positiva per la transizione più recente allo stato attivo.
ROCPosInAlarmCount	DINT	Il numero di volte in cui la condizione di tasso di variazione positiva è stata attivata. Se viene raggiunto il valore massimo, il contatore lascia il valore al valore di conteggio massimo.
ROCNegInAlarmTime	LINT	Registrazione cronologica de momento in cui l'istruzione ALMA ha rilevato che il valore In ha superato il limite della condizione d tasso di variazione negativa per la transizione più recente allo stato attivo.
ROCNegAlarmCount	DINT	Il numero di volte in cui la condizione di tasso di variazione negativa è stata attivata. Se viene raggiunto il valore massimo, il contatore lascia il valore al valore di conteggio massimo.

Parametro di uscita	Tipo di dati	Descrizione
AckTime	LINT	Registrazione cronologica della più recente conferma di condizione. Se l'allarme non richiede conferma, questa registrazione cronologica è uguale all'ora dell'allarme di condizione più recente.
RetToNormalTime	LINT	Registrazione cronologica del ritorno dell'allarme allo stato normale.
AlarmCountResetTime	LINT	Registrazione cronologica che indica quando è stato azzerato il conteggio dell'allarme.
DeliveryER	BOOL	Errore di consegna del messaggio di notifica dell'allarme. Impostato in caso di errore di consegna: non è registrato alcun ricevente degli allarmi o almeno un ricevente registrato non ha ricevuto l'ultimo messaggio di modifica dello stato dell'allarme. Azzerato quando la consegna è avvenuta con successo o è in corso.
DeliveryDN	BOOL	Completamento della consegna del messaggio di notifica dell'allarme. Impostato quando la consegna riesce correttamente: esiste almeno un ricevente registrato e tutti i riceventi registrati hanno ricevuto correttamente l'ultimo messaggio di cambiamento di stato dell'allarme. Azzerato quando la consegna non viene portata a termine correttamente o è in corso.
DeliveryEN	BOOL	Stato di consegna del messaggio di notifica dell'allarme. Impostato quando la consegna è in corso. Azzerato quando la consegna non è in corso.
NoSubscriber	BOOL	L'allarme non aveva riceventi registrati quando ha tentato di consegnare il messaggio più recente. Impostato quando non vi sono riceventi registrati. Azzerato quando esiste almeno un ricevente registrato.
NoConnection	BOOL	I riceventi registrati dell'allarme non erano collegati durante il tentativo di consegnare il messaggio più recente. Impostato quando tutti i riceventi registrati sono scollegati. Azzerato quando almeno un ricevente è collegato o non vi sono riceventi registrati.
CommError	BOOL	Errore di comunicazione nella consegna di un messaggio di allarme. Impostato quando vi sono errori di comunicazione e vengono utilizzati tutti i tentativi. Questo significa che esiste un ricevente registrato con una connessione, ma il controllore non ha ricevuto la conferma della consegna del messaggio. Azzerato quando tutti i riceventi registrati collegati confermano la ricezione del messaggio di allarme.
AlarmBuffered	BOOL	Messaggio di allarme bufferizzato a causa di un errore di comunicazione (CommError è impostato) o connessione interrotta (NoConnection è impostato). Impostato quando il messaggio di allarme è bufferizzato per almeno un ricevente registrato. Azzerato quando il messaggio di allarme non è bufferizzato.
Subscribers	DINT	Numero di riceventi registrati per l'allarme.
SubscNotified	DINT	Numero di riceventi registrati notificati correttamente del cambiamento di stato dell'allarme più recente.
Status	DINT	Indicatori di stato combinati: Status.0 = InstructFault. Status.1 = InFaulted. Status.2 = SeverityInv. Status.3 = AlarmLimitsInv. Status.4 = DeadbandInv. Status.5 = ROCPosLimitInv. Status.6 = ROCNegLimitInv. Status.7 = ROCPeriodInv.

Parametro di uscita	Tipo di dati	Descrizione
InstructFault (Status.0)	BOOL	Esistono condizioni di errore dell'istruzione. Non si tratta di un errore minore o grave del controllore. Verificare gli altri bit di stato per stabilire cosa è accaduto.
InFaulted (Status.1)	BOOL	Il programma utente ha impostato InFault per indicare dati in ingresso di cattiva qualità. L'allarme continua a valutare In per le condizioni di allarme.
SeverityInv (Status.2)	BOOL	La configurazione della gravità dell'allarme non è valida. Se la gravità è <1, l'istruzione usa Severity = 1. Se la gravità è >1000, l'istruzione usa Severity = 1000.
AlarmLimitsInv (Status.3)	BOOL	La configurazione del limite di allarme non è valida (ad esempio, LLimit < LLLimit). Se non è valida, l'istruzione azzera tutti i bit attivi delle condizioni di livello. Finché l'errore non viene azzerato, non è possibile rilevare ulteriori condizioni di livello.
DeadbandInv (Status.4)	BOOL	La configurazione della banda morta non è valida. Se non è valida, l'istruzione usa Deadband = 0.0. Valori validi = 0 ≤ banda morta < intervallo dal primo allarme basso abilitato al primo allarme alto abilitato.
ROCPosLimitInv (Status.5)	BOOL	Limite del tasso di variazione positivo non valido. Se non è valido, l'istruzione usa ROCPosLimit = 0.0, che disabilita il rilevamento del tasso di variazione positivo.
ROCNegLimitInv (Status.6)	BOOL	Limite del tasso di variazione negativo non valido. Se non è valido, l'istruzione usa ROCNegLimit = 0.0, che disabilita il rilevamento del tasso di variazione negativo.
ROCPeriodInv (Status.7)	BOOL	Periodo del tasso di variazione non valido. Se non è valido, l'istruzione usa ROCPeriod = 0.0, che disabilita il rilevamento del tasso di variazione.

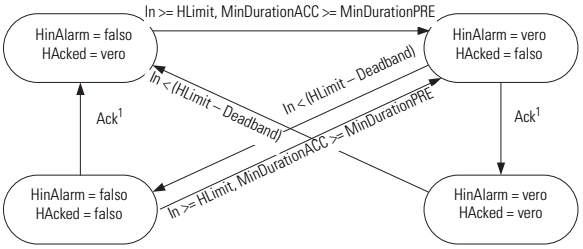
Descrizione

L'istruzione ALMA rileva gli allarmi in base al livello o al tasso di variazione di un valore.

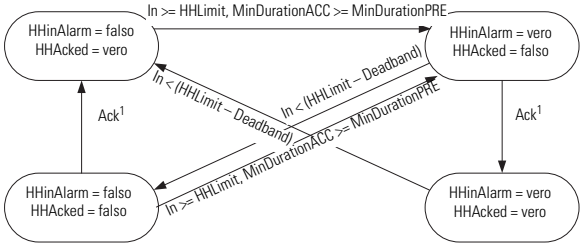
L'istruzione ALMA offre funzioni aggiuntive quando è utilizzata con i software RSLinx Enterprise e FactoryTalk View SE. È possibile visualizzare allarmi nelle visualizzazioni Alarm Summary, Alarm Banner, Alarm Status Explorer e Alarm Log Viewer del software FactoryTalk View SE.

Il software RSLinx Enterprise è ricevente registrato degli allarmi nel controllore. Usare i parametri di uscita per monitorare l'istruzione, vedere lo stato dei riceventi registrati dell'allarme e visualizzare i cambiamenti di stato dell'allarme. Se la connessione al software RSLinx Enterprise si interrompe, il controllore può brevemente bufferizzare i dati di allarme finché la connessione viene ripristinata.

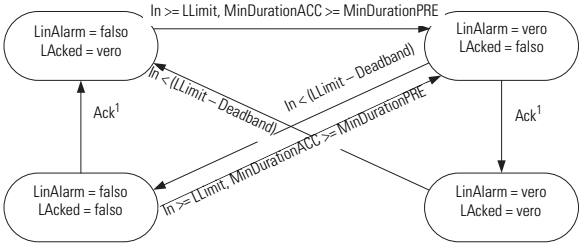
Diagrammi di stato quando è richiesta la conferma di ricezione



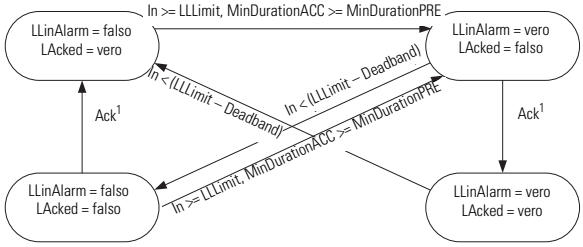
[1] La condizione di allarme alta può essere confermata in numerosi modi diversi: HProgAck, HOperAck, ProgAckAll, OperAckAll, client (RSLogix 5000, software RSView).



[1] La condizione di allarme massima può essere confermata in numerosi modi diversi: HHProgAck, HHOperAck, ProgAckAll, OperAckAll, client (software RSLogix 5000, software RSView).



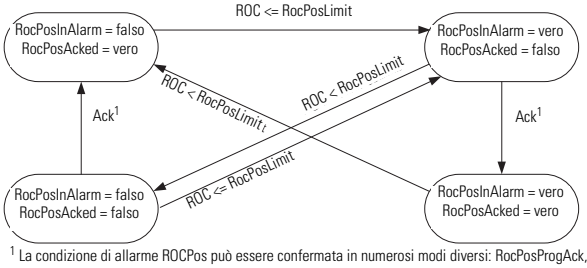
[1] La condizione di allarme alta può essere confermata in numerosi modi diversi: LProgAck, LOperAck, ProgAckAll, OperAckAll, client (software RSLogix 5000, software RSView).



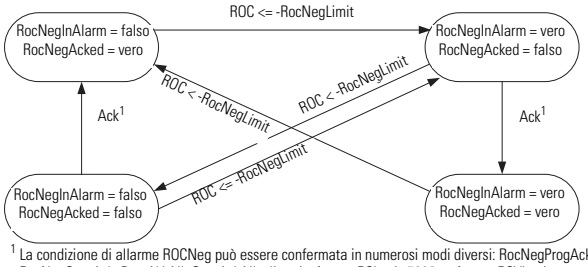
[1] La condizione di allarme minima può essere confermata in numerosi modi diversi: LLProgAck, LLOperAck, ProgAckAll, OperAckAll, client (software RSLogix 5000, software RSView).

$$ROC = \frac{\ln(CurrentSample) - \ln(PreviousSample)}{ROCPeriod}$$

Dove un nuovo campione viene raccolto nella scansione successiva, dopo che ROCPeriod è trascorso.

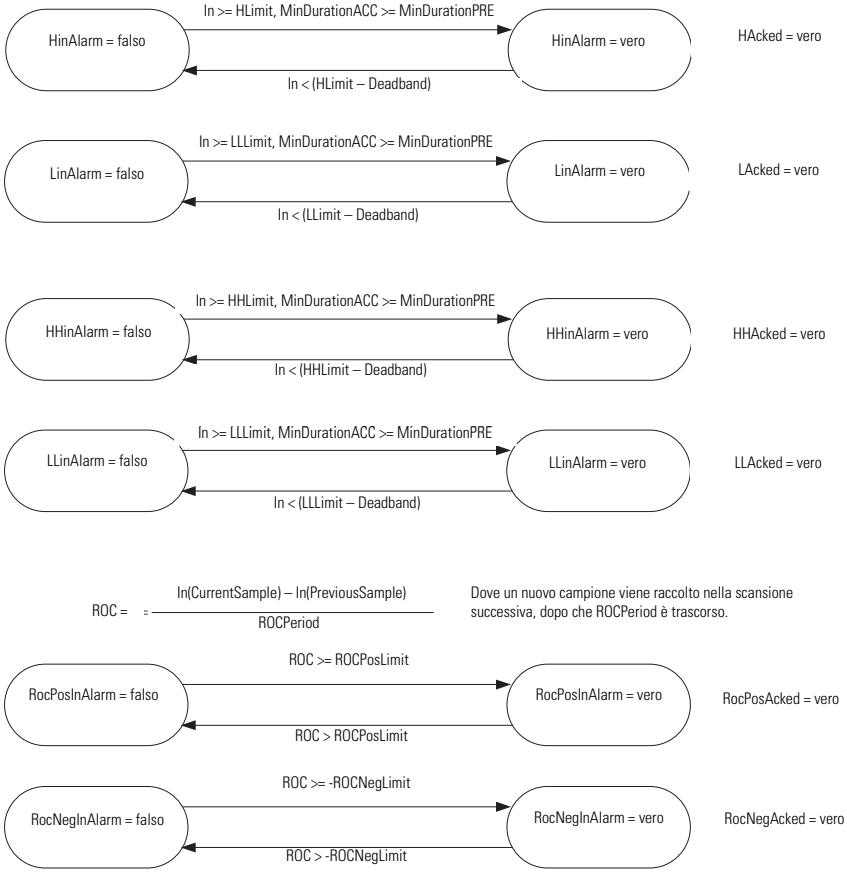


[1] La condizione di allarme ROCPos può essere confermata in numerosi modi diversi: RocPosProgAck, RocPosOperAck, ProgAckAll, OperAckAll, client (software RSLogix 5000, software RSView).



[1] La condizione di allarme ROCNeg può essere confermata in numerosi modi diversi: RocNegProgAck, RocNegOperAck, ProgAkkAll, OperAckAll, client (software RSLogix 5000, software RSView).

Diagrammi di stato quando non è richiesta la conferma di ricezione



Indicatori di stato aritmetico: Per l'uscita ROC vengono impostati gli indicatori di stato aritmetico.

Condizioni di errore:

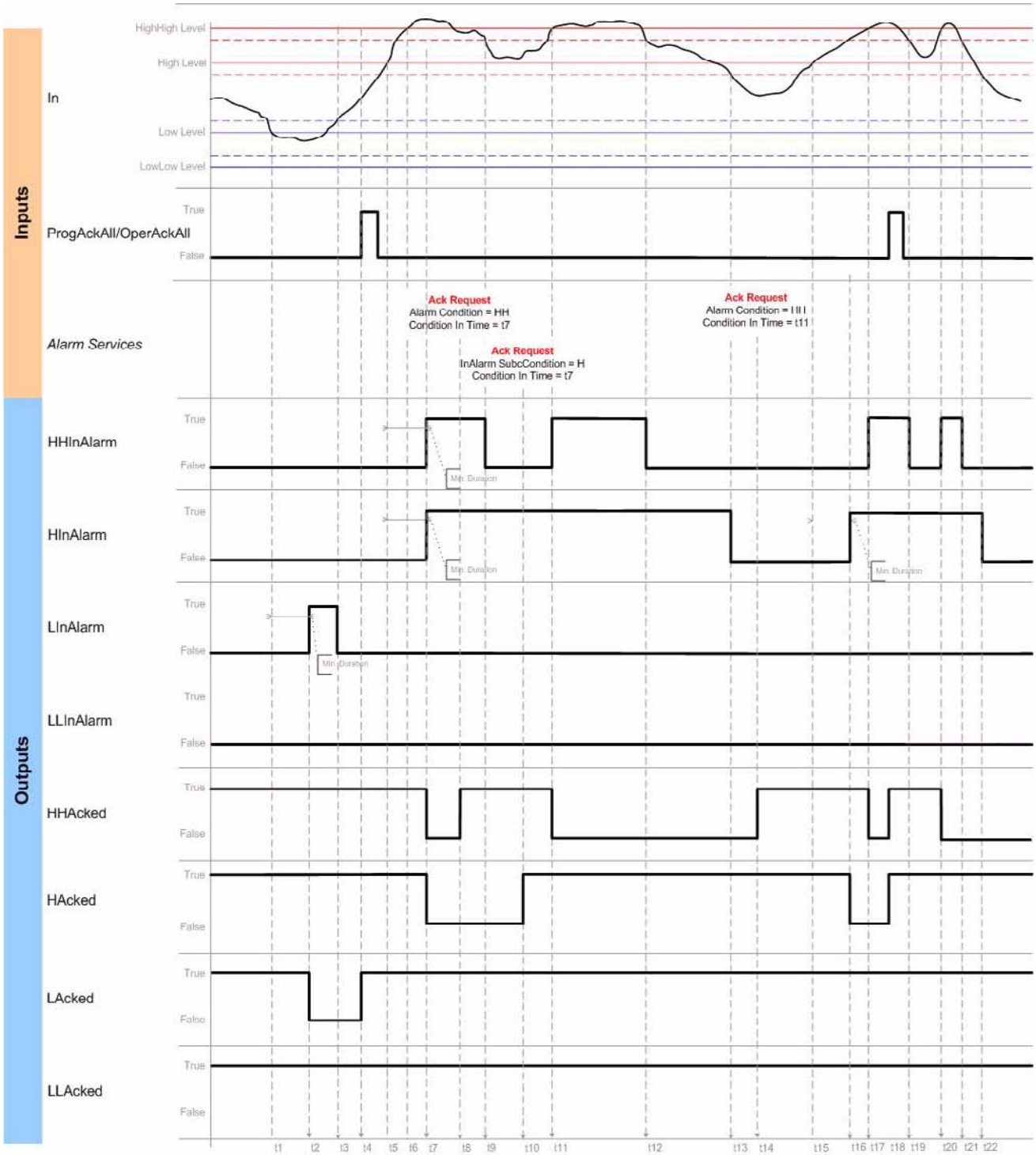
Errore minore:	Tipo errore	Codice errore
Overflow ROC	4	4

Esecuzione:

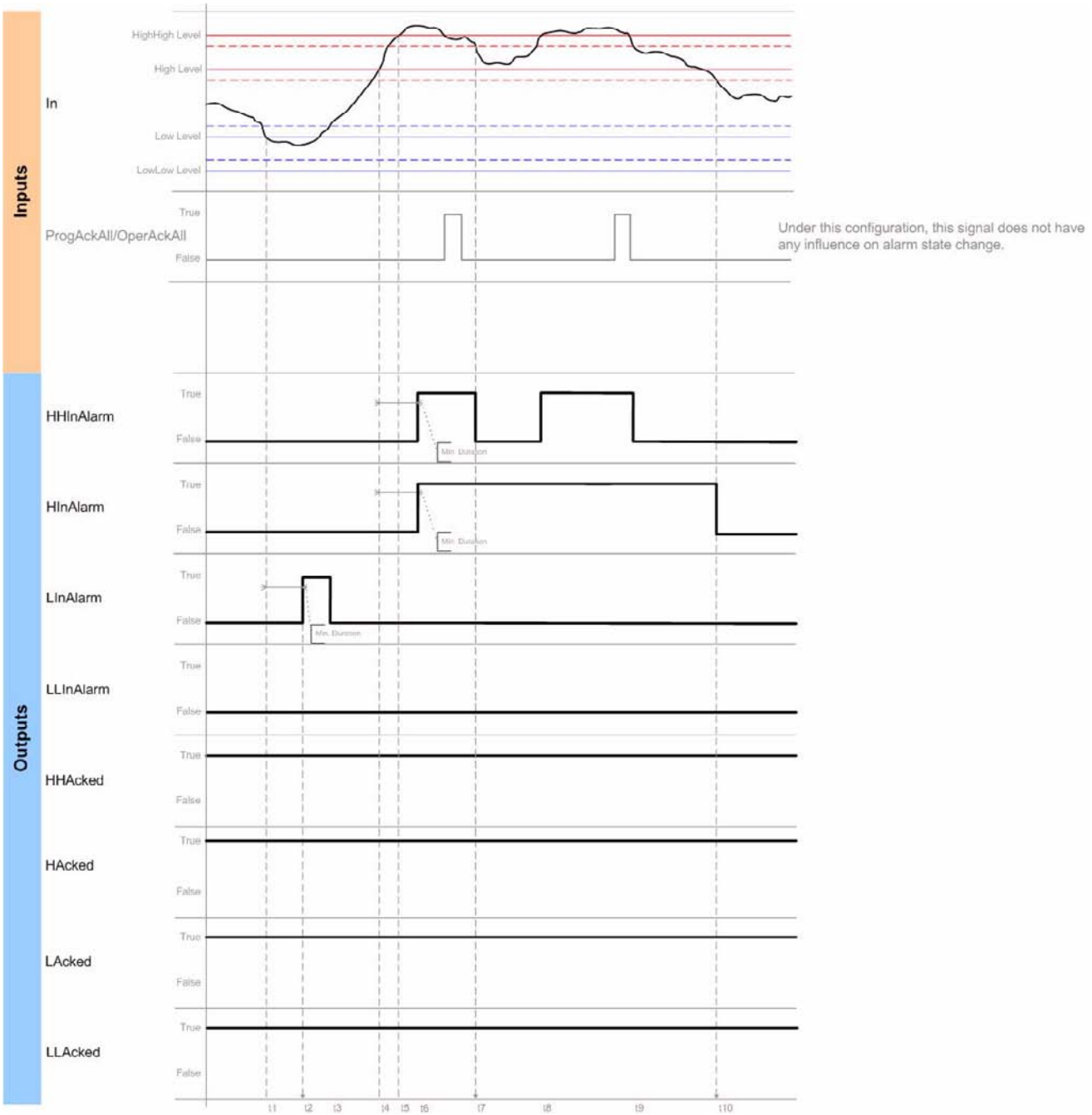
Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso. Tutti i parametri *x*InAlarm vengono azzerati e tutte le condizioni di allarme vengono confermate. Tutte le richieste operatore, le registrazioni cronologiche e gli indicatori di consegna vengono azzerati.
la condizione del ramo di ingresso è falsa	L'istruzione non viene eseguita. EnableOut viene azzerato.
la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Tutte le richieste operatore, le registrazioni cronologiche e gli indicatori di consegna vengono azzerati. Tutti i parametri *x*InAlarm vengono azzerati e tutte le condizioni di allarme vengono confermate.	Tutte le richieste operatore, le registrazioni cronologiche e gli indicatori di consegna vengono azzerati. Tutti i parametri *x*InAlarm vengono azzerati e tutte le condizioni di allarme vengono confermate.
prima scansione dell'istruzione	Nessuna.	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.	Nessuna.
EnableIn viene azzerato	L'istruzione non viene eseguita. EnableOut viene azzerato.	L'istruzione viene eseguita. EnableOut viene sempre impostato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.	L'istruzione viene eseguita. EnableOut viene sempre impostato.
post-scansione	Nessuna.	Nessuna.

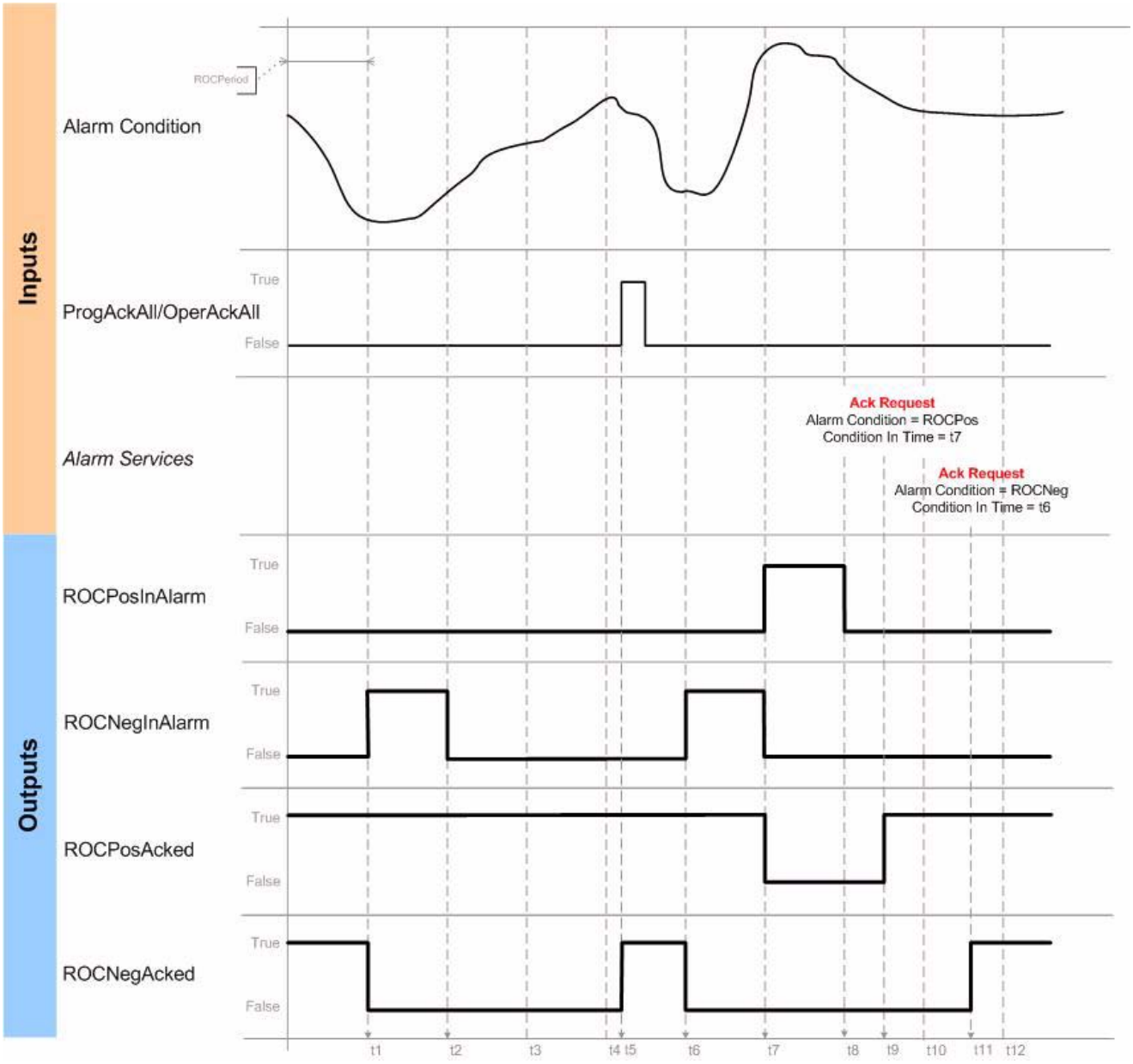
Condizione livello ALMA – Conferma di ricezione richiesta



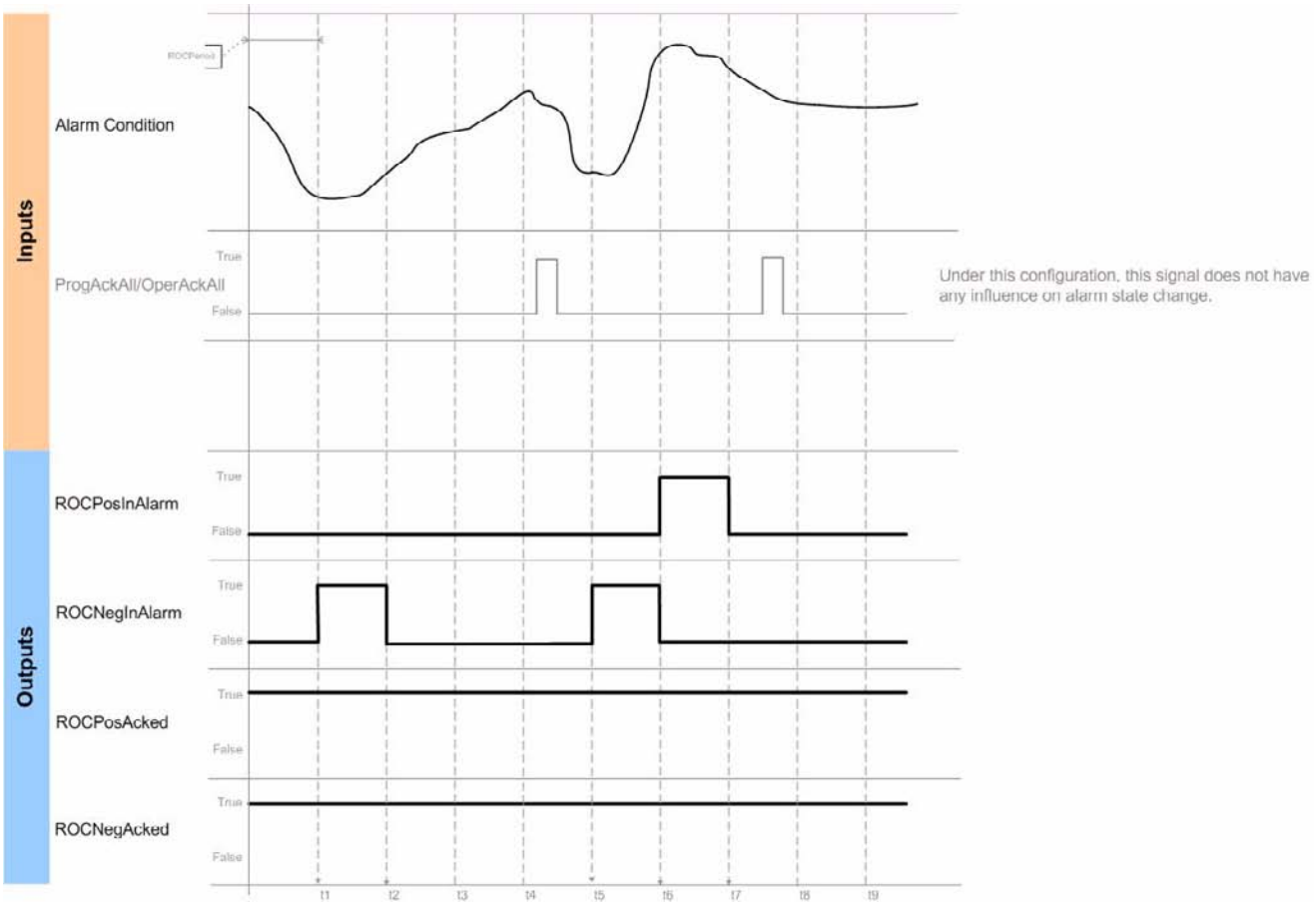
Condizione livello ALMA – Conferma di ricezione non richiesta



Tasso di variazione ALMA – Conferma di ricezione richiesta

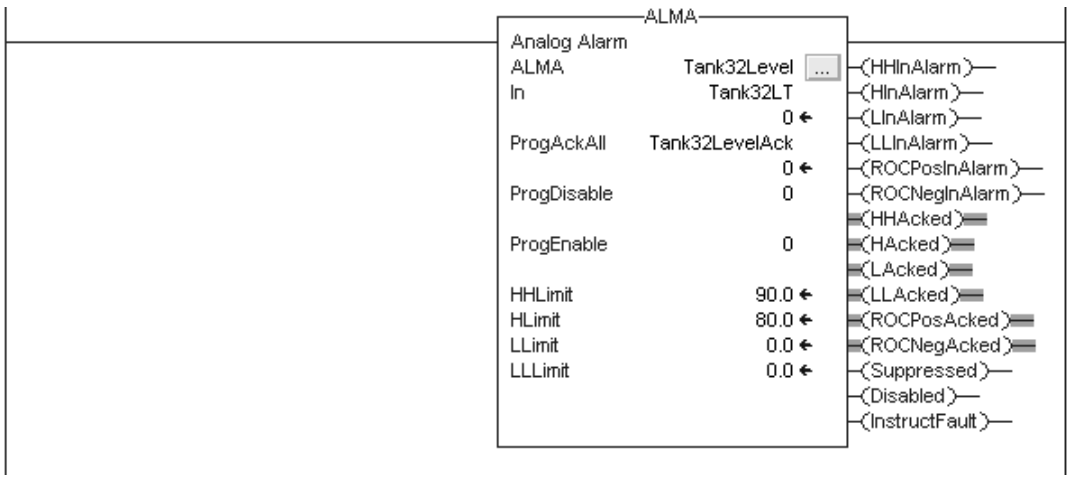


Tasso di variazione ALMA – Conferma di ricezione non richiesta



Esempio: Viene attivato un allarme serbatoio se il livello del serbatoio supera un limite alto o un limite massimo. Confermare via programma la ricezione di tutte le condizioni di allarme con una transizione da azzerato a impostato del valore del tag Tank32LevelAck. La logica applicativa deve azzerare Tank32LevelAck.

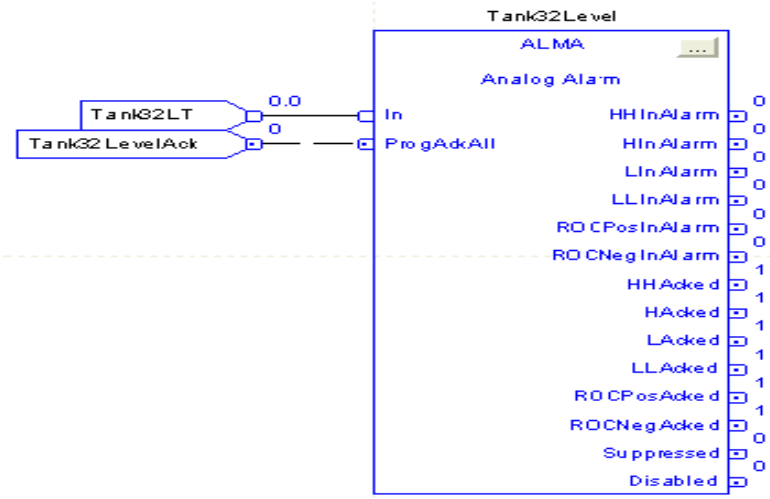
Ladder



Testo strutturato

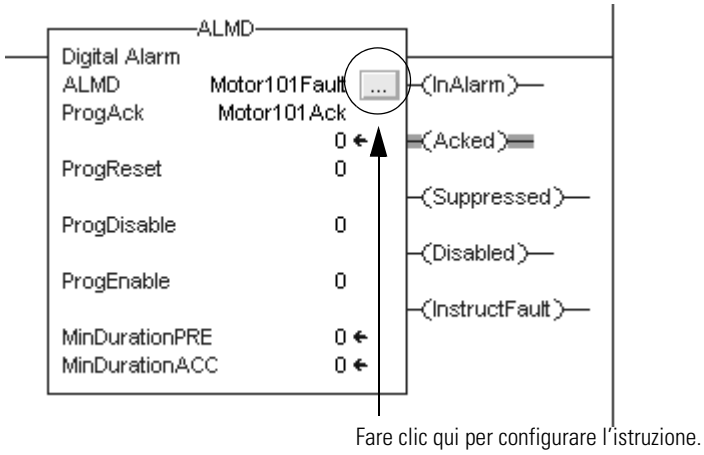
```
ALMA(Tank32Level,Tank32LT,Tank32LevelAck,0, 0);
```

Blocco funzione

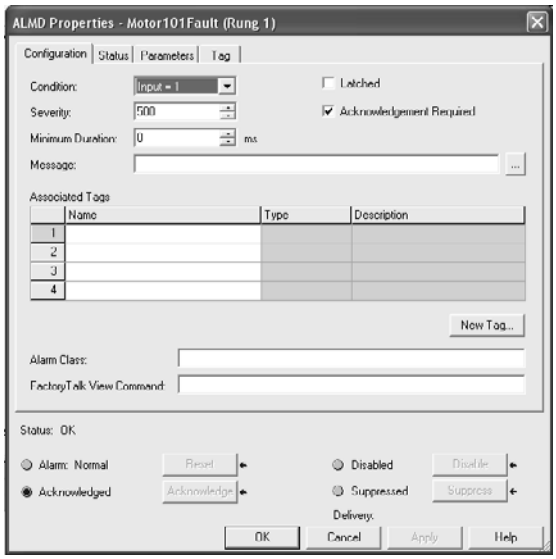


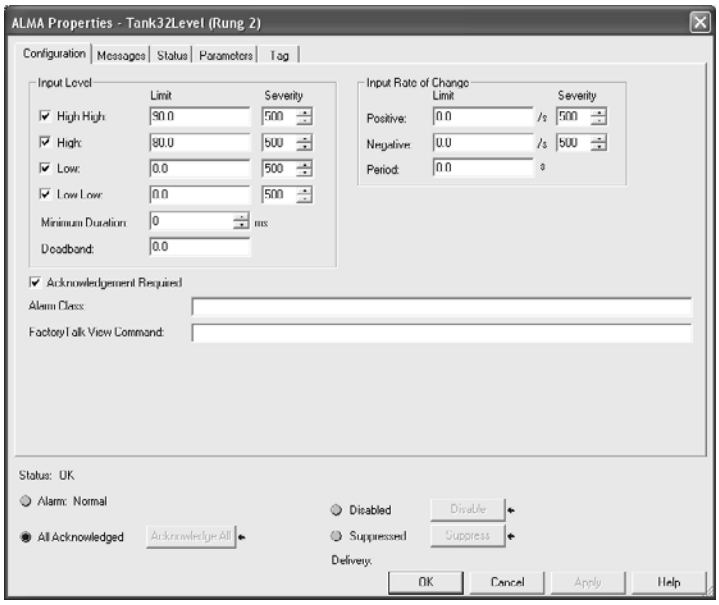
Configurazione di un'istruzione di allarme

Dopo avere immesso l'istruzione ALMD o ALMA e specificato il nome del tag di allarme, utilizzare la finestra di dialogo Alarm Configuration per specificare i dettagli del messaggio.



La finestra di dialogo Properties per l'istruzione di allarme comprende una scheda Configuration.





Per ogni istruzione di allarme, eseguire la configurazione.

Opzione	Descrizione
Condition – ALMD instruction	Condizione per attivare l'allarme. Selezionare Input=1 per un allarme attivo quando In=1. Selezionare Input=0 per un allarme attivo quando In=0.
Input Level – ALMA instruction Input Rate of Change – ALMA instruction	Livello dell'ingresso (massimo, alto, basso o minimo) o tasso di variazione dell'ingresso (positivo o negativo) per attivare l'allarme. Selezionare le condizioni di allarme e immettere i limiti per tali condizioni. Disabilitare le condizioni del tasso di variazione immettendo 0 per il periodo o il limite.
Severity	Selezionare un intervallo di gravità tra 1 e 1000 per indicare l'importanza della condizione di allarme. Una gravità pari a 1 si applica ad allarmi di bassa priorità, una gravità di 1000 indica una condizione di emergenza. Per impostazione predefinita, nel sistema di allarmi ed eventi di FactoryTalk, gli intervalli di gravità sono mappati rispetto alle priorità nel modo seguente: • 1...250 sono priorità basse. • 251...500 sono priorità medie. • 501...750 sono priorità alte. • 751...1000 sono priorità urgenti. È possibile configurare la mappatura gravità-priorità nel sistema di allarmi ed eventi FactoryTalk. Per informazioni dettagliati, consultare la guida di FactoryTalk.
Minimum Duration	Immettere il periodo di tempo minimo in millisecondi per il quale una condizione deve essere attiva prima che sia generato un report dell'allarme.
Latched – ALMD instruction	Selezionare Latched se si desidera che l'allarme resti attivo (InAlarm) dopo che la condizione di allarme torna inattiva (normale). Gli allarmi agganciati richiedono che un comando di azzeramento passi allo stato normale. Il comando di azzeramento deve essere ricevuto dopo che la condizione è tornata normale. I comandi di conferma di ricezione non azzerano un allarme agganciato.

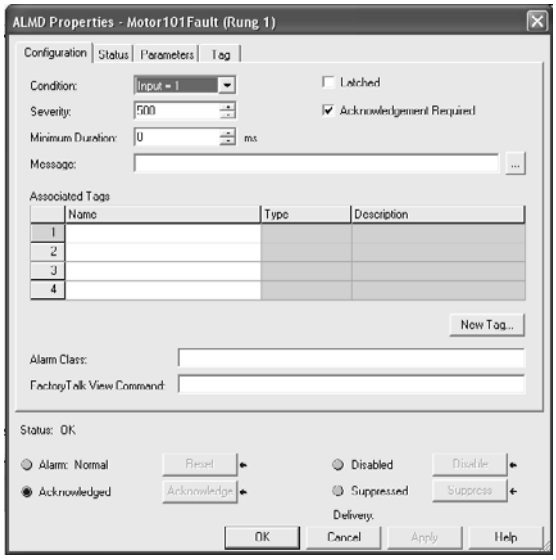
Opzione	Descrizione
Deadband – ALMA instruction	Specificare un valore di banda morta per ridurre il chattering della condizione di allarme causato da piccole fluttuazioni nel valore In. Il valore della banda morta non influenza li limite di allarme per la transizione nello stato attivo, inoltre non è utilizzato durante l'intervallo di durata minima. Una volta che una condizione del livello diventa attiva (InAlarm), rimane tale finché il valore In supera il limite della banda morta specificata. As esempio, se il limite alto è pari a 80, il limite basso è 20 e la banda morta è 5, la condizione alta sarà attiva a ≥ 80 e tornerà in condizione normale a ≤ 75; la condizione bassa sarà attiva a ≤ 20 e tornerà in condizione normale a ≥ ≤ 25. La banda morta non ha alcun effetto sulle condizioni di allarme relative al tasso di variazione.
Acknowledgement Required	Gli allarmi sono configurati per richiedere la conferma di ricezione per impostazione predefinita. La conferma di ricezione indica che un operatore è consapevole della condizione di allarme, a prescindere dal fatto che le condizioni siano tornate allo stato normale. Azzerare l'impostazione Acknowledgement Required quando si desidera che l'allarme compaia e scompaia dal riepilogo degli allarmi sul pannello operatore, senza interazione dell'operatore. Gli allarmi che non richiedono conferma di ricezione hanno sempre lo stato Acked impostato. Se un allarme analogico è configurato come agganciato, anche il comando di azzeramento conferma la ricezione dell'allarme.
Alarm class	Utilizzare la classe dell'allarme per raggruppare allarmi correlati. Specificare esattamente la classe dell'allarme per ciascun allarme che si desidera inserire nella stessa classe. La classe dell'allarme distingue tra caratteri maiuscoli e minuscoli. Ad esempio, specificare Tank Farm A per raggruppare tutti gli allarmi serbatoio per un'area specifica. Oppure, specificare Control Loop per raggruppare tutti gli allarmi degli anelli PID. Sarà quindi possibile visualizzare e filtrare gli allarmi sul pannello operatore in base alla classe a cui appartengono. Ad esempio, un operatore può visualizzare tutti gli allarmi serbatoio o degli anelli PID. La classe dell'allarme non limita gli allarmi per cui un oggetto Alarm Summary si registra per la ricezione. Utilizzare la classe dell'allarme per filtrare gli allarmi visualizzati a un operatore una volta che sono stati ricevuti dall'oggetto Alarm Summary. Il software FactoryTalk View è in grado di filtrare la classe dell'allarme sostituendo i caratteri con caratteri jolly.
View command	Consente di eseguire un comando sulla stazione dell'operatore quando viene richiesto da un operatore per un allarme specifico. Questo permette all'operatore di eseguire qualsiasi comando standard di FactoryTalk, ad esempio richiamare maschere e visualizzazioni specifiche, eseguire macro, accedere a file della guida e lanciare applicazioni esterne. Quando la condizione di allarme si verifica e viene visualizzata all'operatore, un pulsante nelle visualizzazioni del riepilogo e nell'indicatore consente all'operatore di eseguire un comando di visualizzazione associato. Fare attenzione a immettere la sintassi del comando corretta e testare il comando in runtime poiché non viene eseguito alcun controllo per verificare la presenza di errori nel momento in cui il comando viene immesso.

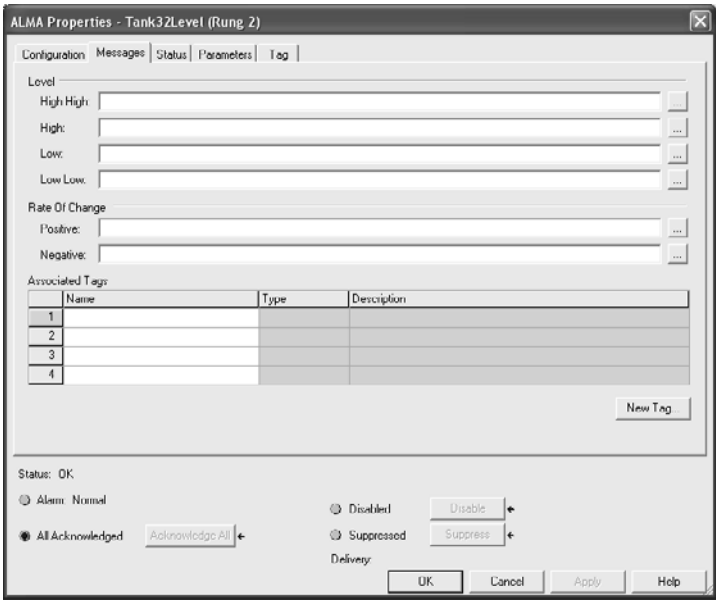
È possibile modificare tutti gli aspetti della configurazione dell'allarme offline e online. Le modifiche online degli allarmi nuovi ed esistenti vengono immediatamente inviate ai riceventi registrati di FactoryTalk (i terminali di pannelli operatori precedenti che stanno semplicemente

interrogando i tag non vengono aggiornati automaticamente). I riceventi registrati di FactoryTalk non devono registrarsi nuovamente per ricevere gli aggiornamenti. Le modifiche online vengono inviate automaticamente dalla struttura di allarme del controllore al resto dell'architettura.

Immissione del testo di un messaggio di allarme

Immettere un testo appropriato per il messaggio da visualizzare quando è attiva una condizione di allarme (InAlarm). Per un'istruzione ALMD, immettere le informazioni sul messaggionella scheda Configuration. Per un'istruzione ALMA, immettere le informazioni sul messaggio nella scheda Message.





Per definire un messaggio di allarme, specificare queste informazioni.

Opzione	Descrizione
Message	La stringa del messaggio contiene le informazioni da visualizzare all'operatore in merito all'allarme. Oltre a immettere del testo, è anche possibile inserire delle variabili. Nell'editor del messaggio, selezionare la variabile desiderata e aggiungerla in qualsiasi punto della stringa del messaggio. La stringa del messaggio può contenere un massimo di 255 caratteri, compresi i caratteri che specificano le variabili integrate (non il numero di caratteri nei valori effettivi delle variabili integrate). Ad esempio, /*S:0 %Tag1*/ specifica un tag di una stringa e aggiunge 13 caratteri alla lunghezza della stringa, ma il valore effettivo del tag della stringa potrebbe contenere 82 caratteri. Non è possibile accedere via programma alla stringa del messaggio di allarme per il tag di allarme. Per modificare il messaggio di allarme in base a eventi specifici, configurare uno dei tag associati come tipo di dati stringa e integrare il tag associato nel messaggio. È possibile avere più versioni dei messaggi in diverse lingue. La lingua diversa può essere immessa con l'utility di importazione/esportazione. Per ulteriori informazioni, consultare pagina 66.

Opzione	Descrizione
Associated tags	È possibile selezionare fino a quattro tag aggiuntivi dal progetto del controllore per associarli all'allarme. I valori di tali tag sono inviati con un messaggio di allarme al server degli allarmi. Ad esempio, un allarme digitale per una valvola regolatrice di pressione può anche comprendere informazioni come la velocità della pompa e la temperatura del serbatoio. I tag associati possono essere tipi di dati di tipo elementare (BOOL, DINT, INT, SINT o REAL) o una stringa. Gli elementi possono essere in un UDT o in un array. Non sono consentiti riferimenti di array di variabili. Se l'allarme è nell'ambito del controllore, anche i tag associati devono essere nell'ambito del controllore. Opzionalmente, integrare i tag associati nella stringa del testo del messaggio. I valori dei tag associati sono sempre inviati con l'allarme, visibile dall'operatore, e immessi nel log cronologico, a prescindere dal fatto che siano integrati nella stringa del messaggio.

Variabili della stringa del messaggio

All'interno della stringa di un messaggio è possibile integrare le seguenti informazioni variabili.

Variabile	Integra nella stringa del messaggio	Codice predefinito aggiunto alla stringa del messaggio
Alarm name	Il nome dell'allarme, composto da nome del controllore, nome del programma e nome del tag. Ad esempio, [Zone1Controller]Program:Main.MyAlarmTagName.	`/*S:0 %AlarmName*/`
Condition name	La condizione che attiva l'allarme: • l'allarme digitale visualizza l'intervento • l'allarme analogico visualizza HiHi, Hi, Lo, LoLo, ROC_POS o ROC_NEG.	`/*S:0 %ConditionName*/`
Input value	Il valore di ingresso nell'allarme: • l'allarme digitale visualizza 0 o 1 • l'allarme analogico visualizza il valore della variabile di ingresso monitorata dall'allarme.	`/*N:5 %InputValue NOFILL DP:0*/`
Limit value	La soglia dell'allarme: • l'allarme digitale visualizza 0 o 1 • l'allarme analogico visualizza il controllo dell'intervallo effettivamente configurato per la condizione dell'allarme analogico.	`/*N:5 %LimitValue NOFILL DP:0*/`
Severity	La gravità configurata della condizione di allarme.	`/*N:5 %Severity NOFILL DP:0*/`
Values of associated tags	Il valore di un tag configurato da includere nell'evento allarme.	`/*N:5 %Tag1 NOFILL DP:0*/`

Il codice dipende dal tipo di tag selezionato, dal numero di cifre o caratteri contenuti nel valore di un tag e dal fatto che si desideri riempire a sinistra i bit vuoti con spazi o zeri. Ad esempio:

Tag	Codice
Valore BOOL	`/*N:1 %Tag1 NOFILL DP:0*/`
Valore DINT, 9 cifre, riempimento a sinistra con spazi	`/*N:9 %Tag2 SPACEFILL DP:0*/`
Valore di ingresso REAL, 9 cifre (compreso il decimale), 3 cifre decimali, riempimento a sinistra con zeri	`/*N:9 %InputValue NOFILL DP:3*/`
Valore REAL, 8 cifre (compreso il decimale), 4 cifre decimali, riempimento a sinistra con zeri	`/*N:8 %Tag3 ZEROFILL DP:4*/`
Valore stringa, nessuna larghezza fissa	`/*S:0 %Tag4*/`
Valore stringa, 26 caratteri, larghezza fissa	`/*S:26 %Tag4*/`

Tutte queste informazioni variabili sono comprese nei dati dell'allarme, visibili dall'operatore, e immesse nel registro cronologico, a prescindere dal fatto che siano integrate nel testo del messaggio.

Versioni in più lingue dei messaggi di allarme

È possibile mantenere i messaggi di allarme in più lingue. Immettere le diverse lingue nelle versioni in lingua associate del software di programmazione RSLogix 5000 o in un file di importazione/esportazione (.CSV o .TXT).

È possibile accedere al testo dei messaggi di allarme da un file di importazione/esportazione (.CSV o .TXT). e aggiungere righe ulteriori per le versioni tradotte della stringa di messaggio originale. I messaggi in più lingue usano i codici linguistici ISO nella colonna TYPE. Il testo del messaggio di allarme, compresi i codici delle variabili integrate, per l'operatore si trova nella colonna DESCRIPTION. SPECIFIER identifica la condizione di allarme.

TYPE	SCOPE	NAME	DESCRIPTION	DATATYPE	SPECIFIER
TAG		alma1		ALARM_ANALOG	
ALMMSG:en-us		alma1	HH alarm for operator in English		HH
ALMMSG:en-us		alma1	H alarm for operator in English		H
ALMMSG:en-us		alma1	L alarm for operator in English		L
ALMMSG:en-us		alma1	LL alarm for operator in English		LL
ALMMSG:en-us		alma1	ROC positive alarm for operator in English		POS
ALMMSG:en-us		alma1	ROC positive alarm for operator in English		NEG
ALMMSG:de-ch		alma1	HH Mitteilung für den Operator auf Deutsch		HH
ALMMSG:de-ch		alma1	H Mitteilung für den Operator auf Deutsch		H
TAG		almd1		ALARM_DIGITAL	
ALMMSG:en-us		almd1	digital alarm for operator in English		AM

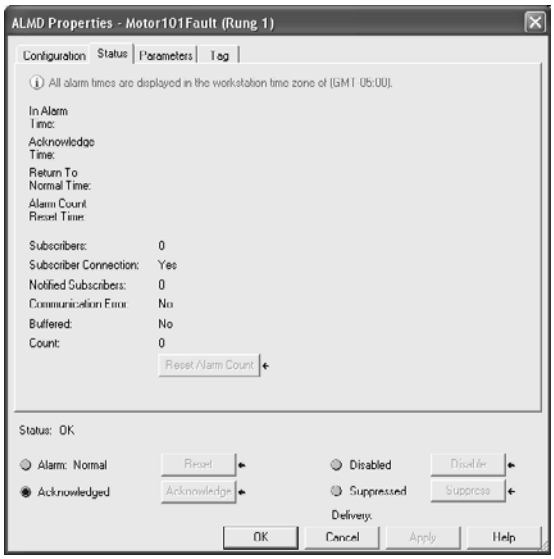
Usare l'utility di importazione/esportazione per creare e tradurre stringhe di messaggi in più lingue. Il formato di importazione/esportazione .TXT supporta i caratteri a doppio byte, pertanto è possibile utilizzare questo formato per tutte le lingue, comprese

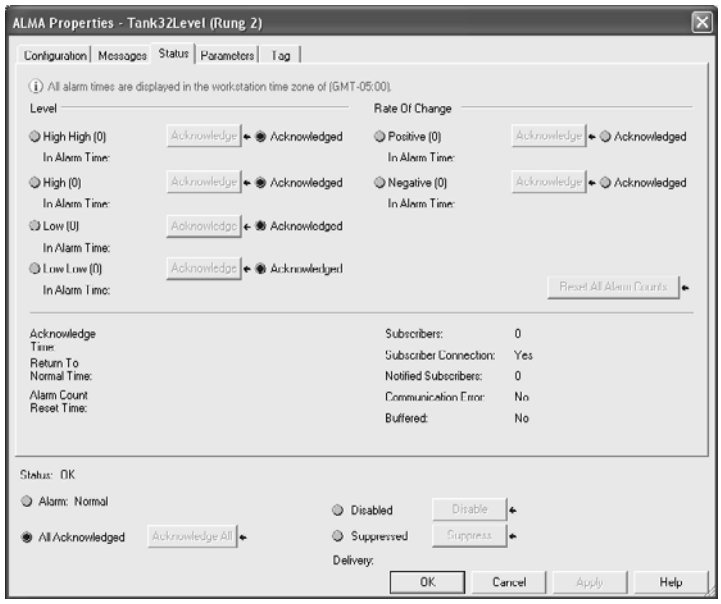
cinese, giapponese e coreano. Il formato di importazione/esportazione .CSV non supporta i caratteri a doppio byte.

L'importazione ed esportazione di messaggi esegue sempre un'unione. Se si cancella un messaggio in un file .CSV o .TXT, il messaggio non viene cancellato dal file .ACD. Per cancellare un messaggio, importare il file .CSV o .TXT con i campi Type, Name e Specifier compilati ma senza alcuna descrizione.

Monitoraggio dello stato degli allarmi

Nella scheda Status della finestra di dialogo dell'allarme, è possibile monitorare la condizione di allarme, confermare la ricezione di un allarme, disabilitare, sopprimere o azzerare un allarme. Usare le selezioni della finestra di dialogo per vedere come si comporta un allarme, senza che occorra un pannello operatore operativo.





Bufferizzazione degli allarmi

Per ricevere messaggi di allarme basati sul controllore, i client degli allarmi (come un server RSLinx Enterprise) devono registrarsi come riceventi degli allarmi nel controllore Logix. Il controllore mantiene una connessione con ciascun ricevente registrato e monitora lo stato di tale connessione.

Quando si verifica un cambiamento dello stato di un allarme, le istruzioni di allarme nel controllore mettono in cache le informazioni necessarie (come le registrazioni cronologiche e i valori dei tag associati) e richiedono la trasmissione di un messaggio di allarme a tutti i riceventi registrati. Il meccanismo di pubblicazione invia i messaggi di allarme a ogni ricevente registrato il prima possibile.

Se un ricevente registrato non conferma la ricezione del messaggio di allarme o se la connessione a un ricevente registrato non funziona, il

meccanismo di pubblicazione memorizza i messaggi di allarme non consegnati in un buffer di 100 KB. Ogni ricevente registrato ha il proprio buffer, quindi eventuali problemi di comunicazione con un ricevente registrato non interferiscono con l'invio ad altri riceventi. Quando il buffer è pieno, i messaggi di allarme più recenti vengono ignorati. Il buffer viene creato quando il ricevente registrato stabilisce una connessione iniziale, mantenuta per un periodo di tempo configurabile (da 0 a 120 minuti, l'impostazione predefinita è 20 minuti) dopo che il ricevente registrato perde la connessione.

Quando il ricevente registrato stabilisce nuovamente una connessione nell'intervallo di timeout del buffer, riceve lo stato corrente di tutti gli allarmi, inizia a ricevere i messaggi di allarme correnti e carica eventuali messaggi bufferizzati accumulati. Anche nel caso in cui il buffer fosse pieno e siano stati ignorati dei messaggi, il ricevente registrato sincronizza accuratamente lo stato corrente degli allarmi (comprese le registrazioni cronologiche più recenti InAlarmTime, RetToNormalTime e AckTime).

Il buffer continua finché è pieno. Una volta riempito, il buffer smette di aggiungere transizioni di allarme finché il ricevente registrato non libera dello spazio del buffer.

Accesso via programma alle informazioni di allarme

Ogni istruzione di allarme ha una struttura di allarme che memorizza le informazioni su configurazione ed esecuzione dell'allarme. La struttura dell'allarme comprende elementi di controllo sia del programma sia dell'operatore. Le istruzioni di allarme non usano le impostazioni di modalità per determinare se è attivo l'accesso programma o l'accesso operatore, pertanto tali elementi sono sempre attivi.

Esistono tre modi per eseguire le azioni su un'istruzione di allarme.

Accesso	Elementi della struttura di allarme	Considerazioni
Programma utente	• ProgAck • ProgReset • ProgSuppress • ProgDisable • ProgEnable	Usare la logica del controllore per accedere via programma agli elementi del sistema di allarme. Ad esempio, il programma di controllo può determinare se disabilitare una serie di allarmi correlati a un'unica causa radice. Il programma di controllo potrebbe, ad esempio, disabilitare un'istruzione di allarme (MyDigitalAlarm del tipo di dati ALARM_DIGITAL) accedendo al membro del tag MyDigitalAlarm.ProgDisable.

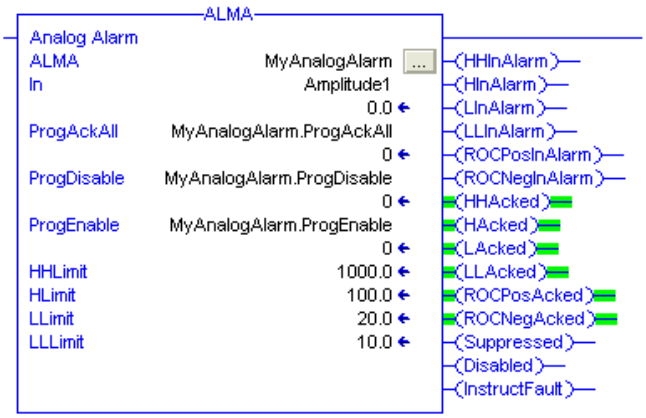
Accesso	Elementi della struttura di allarme	Considerazioni
Pannello operatore personalizzato	• OperAck • OperReset • OperSuppress • OperDisable • OperEnable	Creare una maschera del pannello operatore per accedere agli elementi del sistema di allarme. Ad esempio, se l'operatore deve rimuovere uno strumento, invece di disabilitare o sopprimere manualmente e singolarmente gli allarmi nella schermata degli allarmi, l'operatore può premere un tasto di disabilitazione che accede al tag MyDigitalAlarm.OperDisable. I parametri operatore funzionano con qualsiasi interfaccia di Rockwell Automation o di terze parti per consentire il controllo degli stati di allarme. Quando un parametro operatore è impostato, l'istruzione valuta se è in grado di rispondere alla richiesta, quindi azzera sempre il parametro.
Oggetto pannello operatore standard	Non accessibile	La normale interazione con l'operatore avviene attraverso il riepilogo degli allarmi, l'indicatore di allarme e il navigatore degli stati di allarme nell'applicazione FactoryTalk View. Questa interazione è simile all'opzione di pannello operatore sopra descritta, ma non vi è alcuna visibilità o interazione programmatica.

Quando si crea un'istruzione di allarme, occorre creare e assegnare un tag del tipo di dati corretto per quell'allarme. Ad esempio, creare MyDigitalAlarm con il tipo di dati ALARM_DIGITAL. Nella logica ladder questi parametri di istruzione devono essere immessi nell'istruzione:

- ProgAck
- ProgReset
- ProgDisable
- ProgEnable

In logica ladder e in testo strutturato, il valore o il tag assegnato a un parametro di istruzione (come ProgAck) viene automaticamente scritto nel membro del tag di allarme (come MyAnalogAlarm.ProgAck) ogni volta che l'istruzione viene scansionata.

In logica ladder e in testo strutturato, se si desidera accedere programmaticamente alla struttura di allarme, assegnare il tag di struttura al parametro nell'istruzione. Ad esempio, per usare MyAnalogAlarm.ProgAck nella logica, assegnare il tag MyAnalogAlarm.ProgAck al parametro ProgAck.



Soppressione o disabilitazione degli allarmi

Sopprimere gli allarmi per rimuovere dal pannello operatore gli allarmi di cui si conosce l'esistenza, pur mantenendoli. Questo consente di svuotare il riepilogo allarmi mentre si sta risolvendo un allarme noto senza continuare a vedere le informazioni sugli allarmi. Un allarme soppresso non compare nel riepilogo allarmi dell'operatore o nelle schermate degli indicatori, ma continua a essere inviato ai riceventi registrati, registrato nel database storico, è ancora in grado di cambiare stato di allarme, riceve una registrazione cronologica e reagisce ad altre interazioni programmatiche o con l'operatore.

- Quando un allarme è soppresso continua a funzionare normalmente, monitorare il parametro In per informazioni sulle condizioni di allarme e a rispondere alle richieste di conferma. Tutti i riceventi registrati vengono informati di questo evento ed eventuali messaggi di allarme generati mentre l'allarme è in stato soppresso comprendono lo stato soppresso. I client degli allarmi possono rispondere in modo diverso agli allarmi soppressi. Ad esempio, gli allarmi soppressi possono essere registrati nel database storico ma non annunciati all'operatore.
- Quando la soppressione di un allarme viene eliminata, ne vengono notificati tutti i riceventi registrati e i messaggi che vengono loro inviati non contengono più lo stato soppresso.

Disabilitare un allarme per trattare l'allarme come se non esistesse nel programma di controllo. Un allarme disabilitato non cambia stato di allarme né viene registrato nel database storico. Viene comunque tenuta traccia di un allarme disabilitato, che può essere nuovamente abilitato in Alarm Status Explorer nel software FactoryTalk View SE.

- Quando un allarme è disabilitato, tutte le condizioni sono impostate nello stato iniziale (InAlarm è azzerato e Acked è impostato). Il parametro In non è monitorato per quanto riguarda le condizioni di allarme. Tutti i riceventi registrati vengono notificati di questo evento.
- Quando un allarme è abilitato inizia a monitorare il parametro In per verificare eventuali condizioni di allarme. Tutti i riceventi registrati vengono notificati di questo evento.

Esecuzione degli allarmi del controllore

Gli allarmi del controllore elaborano gli ingressi da due sorgenti.

Sorgente	Descrizione
Membri del tag di allarme	I membri del tag di allarme sono elaborati, per la maggior parte, quando l'applicazione utente scansiona l'istruzione di allarme. Questo comprende: • elaborazione delle modifiche ai parametri di configurazione. • valutazione della condizione di allarme. • misurazione del tempo trascorso per MinDuration. • rilevamento delle registrazioni cronologiche InAlarmTime e RetToNormalTime. • rilevamento dei valori di tag associati. • elaborazione dei comandi Prog e Oper. Inoltre, questi membri di stato del tag di allarme vengono aggiornati man mano che i messaggi di allarme vengono consegnati a ciascun ricevente registrato, in modo asincrono rispetto alla scansione del programma: • DeliveryEN, DeliveryER, DeliveryDN • NoSubscriber, NoConnection, CommError, AlarmBuffered, SubscNotified
Messaggi client	I messaggi del client vengono elaborati man mano che vengono ricevuti, in modo asincrono rispetto alla scansione del programma. • Comandi Reset, Acknowledge, Disable/Enablee Suppress/Unsuppress da un terminale RSLogix 5000. • Comandi Reset, Acknowledge, Disable/Enable e Suppress/Unsuppress da un ricevente registrato di allarmi FactoryTalk View SE

Prestare attenzione quando si decide dove inserire le istruzioni di allarme nell'applicazione. La precisione delle registrazioni cronologiche dipende dalla velocità con cui l'istruzione viene scansionata dopo che la condizione di allarme cambia stato. L'accumulo di tempo MinDuration e i calcoli del tasso di variazione richiedono scansioni ripetute entro intervalli temporali definiti dall'applicazione dell'utente. Le istruzioni di allarme devono continuare a essere scansionate dopo che la condizione di allarme diventa falsa, in modo che possa essere rilevata la transizione ReturnToNormal. Ad esempio, se si desidera una precisione di 10 ms sulle registrazioni cronologiche, è possibile inserire le istruzioni di allarme che hanno bisogno di quella risoluzione in un task periodico di 10 ms.

Utilizzo della memoria del controllore

Come linea guida, usare le seguenti dimensioni degli allarmi per calcolare grossolanamente l'utilizzo della memoria del controllore:

- tipicamente 1 KB per allarme digitale senza tag associati

Esempio di allarme digitale	Dimensione approssimativa
Allarme digitale senza tag associati e questa configurazione: • Messaggio di allarme: Contactor Fault • Classe allarme: Tank Farm A	1012 byte
Allarme digitale con due tag associati e questa configurazione: • Messaggio di allarme: Contactor Fault • Classe allarme: Tank Farm A • Tag associato 1 = tipo di dati DINT • Tag associato 2 = tipo di dati DINT	1100 byte
Allarme digitale con due tag associati e questa configurazione: • Messaggio di allarme: Contactor Fault • Classe allarme: Tank Farm A • Tag associato 1 = tipo di dati DINT • Tag associato 2 = tipo di dati STRING	1522 byte

- tipicamente 2.2 KB per allarme analogico senza tag associati

Esempio di allarme analogico	Dimensione approssimativa
Allarme analogico senza tag associati e questa configurazione: • Messaggio di allarme massimo: Level Alarm • Messaggio di allarme alto: Level Alarm • Messaggio di allarme basso: Level Alarm • Messaggio di allarme minimo: Level Alarm • Messaggio tasso di variazione positivo: Fill Too Fast • Messaggio tasso di variazione negativo: Empty Too Fast • Classe allarme: Tank Farm A	2228 byte
Allarme analogico con due tag associati e questa configurazione: • Messaggio di allarme massimo: Level Alarm • Messaggio di allarme alto: Level Alarm • Messaggio di allarme basso: Level Alarm • Messaggio di allarme minimo: Level Alarm • Messaggio tasso di variazione positivo: Fill Too Fast • Messaggio tasso di variazione negativo: Empty Too Fast • Classe allarme: Tank Farm A • Tag associato 1 = tipo di dati DINT • Tag associato 2 = tipo di dati DINT	2604 byte

Esempio di allarme analogico	Dimensione approssimativa
Allarme analogico con due tag associati e questa configurazione: • Messaggio di allarme massimo: Level Alarm • Messaggio di allarme alto: Level Alarm • Messaggio di allarme basso: Level Alarm • Messaggio di allarme minimo: Level Alarm • Messaggio tasso di variazione positivo: Fill Too Fast • Messaggio tasso di variazione negativo: Empty Too Fast • Classe allarme: Tank Farm A • Tag associato 1 = tipo di dati DINT • Tag associato 2 = tipo di dati STRING	4536 byte

Stringhe di messaggio più lunghe, così come stringhe di messaggio in più lingue, consumano ulteriore memoria del controllore.

L'effettivo utilizzo della memoria dipende dal modo in cui l'allarme è configurato, dalla lunghezza del messaggio e da eventuali tag associati passati con l'allarme.

Tempo di scansione

Questi tempi d'esecuzione mostrano in che modo le istruzioni ALMD e ALMA incidono sul tempo di scansione totale.

Stato del ramo		Tempo d'esecuzione	
		Digital Alarm (ALMD)	Analog Alarm (ALMA)
Nessun cambiamento dello stato dell'allarme	Ramo falso	8 µs	17 µs
	Ramo vero	8 µs	60 µs
Cambiamento dello stato dell'allarme	Ramo falso	35 µs	17 µs
	Ramo vero	35 µs	126 µs

Un cambiamento di stato dell'allarme è un evento che cambia la condizione dell'allarme, ad esempio la conferma di ricezione o la soppressione dell'allarme. Ridurre al minimo il potenziale di cambiamento di stato contemporaneo di molti allarmi creando dipendenze tra gli allarmi correlati. Cambiamenti contemporanei eccessivi possono incidere significativamente sul tempo di scansione del codice dell'applicazione.

Note:

Istruzioni di bit

(XIC, XIO, OTE, OTL, OTU, ONS, OSR, OSF, OSRI, OSFI)

Introduzione

Usare le istruzioni di bit (tipo relè) per monitorare e controllare lo stato dei bit.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
abilitare le uscite quando un bit viene impostato	XIC	ladder testo strutturato(1)	76
abilitare le uscite quando un bit viene azzerato	XIO	ladder testo strutturato(1)	78
impostare un bit	OTE	ladder testo strutturato(1)	80
impostare un bit (ritentivo)	OTL	ladder testo strutturato(1)	82
azzerare un bit (ritentivo)	OTU	ladder testo strutturato(1)	84
abilitare le uscite per una scansione ogni volta che un ramo diventa vero	ONS	ladder testo strutturato(1)	86
impostare un bit per una scansione ogni volta che un ramo diventa vero	OSR	ladder	89
impostare un bit per una scansione ogni volta che un ramo diventa falso	OSF	ladder	92
impostare un bit per una scansione ogni volta che il bit di ingresso viene impostato nel blocco funzione	OSRI	testo strutturato blocco funzione	94
impostare un bit per una scansione ogni volta che il bit di ingresso viene azzerato nel blocco funzione	OSFI	testo strutturato blocco funzione	97

(1) Non esiste un'istruzione equivalente in testo strutturato. Utilizzare un altro tipo di programmazione in testo strutturato per ottenere lo stesso risultato. Vedere la descrizione dell'istruzione.

Examine If Closed (XIC)

L'istruzione XIC esamina il bit di dati per verificare se è impostato.

Operandi:

Ladder

Operando	Tipo	Formato	Descrizione
bit di dati	BOOL	tag	bit da testare

Testo strutturato

Il testo strutturato non ha un'istruzione XIC, ma è possibile ottenere gli stessi risultati usando un costrutto IF...THEN.

```
IF bit_dati THEN
    <istruzione>;
END_IF;
```

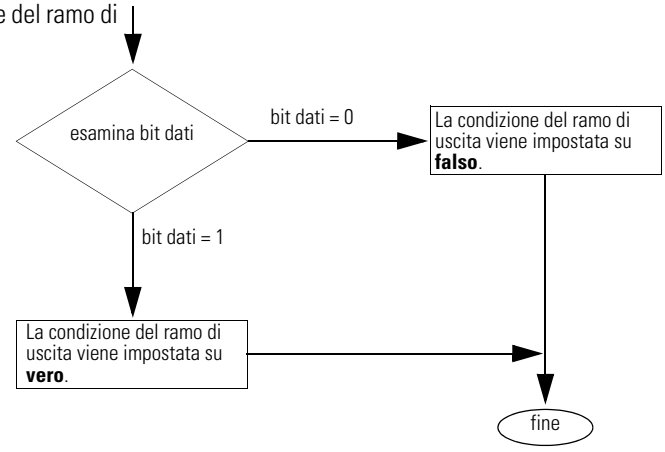
Consultare l'Appendice B, Attributi di un blocco funzione per informazioni sulla sintassi dei costrutti nel testo strutturato.

Descrizione: L'istruzione XIC esamina il bit di dati per verificare se è impostato.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

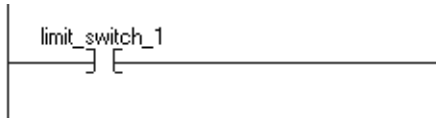
Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

esamina bit dati
bit dati = 0
La condizione del ramo di uscita viene impostata su **falso**.
bit dati = 1
La condizione del ramo di uscita viene impostata su **vero**.
fine

Esempio 1: Se *limit_switch_1* è impostato, viene abilitata l'istruzione successiva (la condizione del ramo di uscita è vera).

Ladder

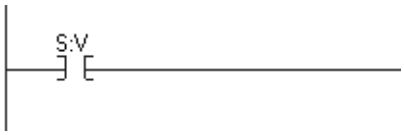


Testo strutturato

```
IF limit_switch THEN
    <istruzione>;
END_IF;
```

Esempio 2: Se *S:V* è impostato (a indicare che si è verificato un overflow), viene abilitata l'istruzione successiva (la condizione del ramo di uscita è vera).

Ladder



Testo strutturato

```
IF S:V THEN
    <istruzione>;
END_IF;
```

Examine If Open (XIO)

L'istruzione XIO esamina il bit di dati per verificare se è azzerato.

Operandi:

Ladder

Operando	Tipo	Formato	Descrizione
bit di dati	BOOL	tag	bit da testare

Testo strutturato

Il testo strutturato non ha un'istruzione XIO, ma è possibile ottenere gli stessi risultati usando un costrutto IF...THEN.

```
IF NOT bit_dati THEN
    <istruzione>;
END_IF;
```

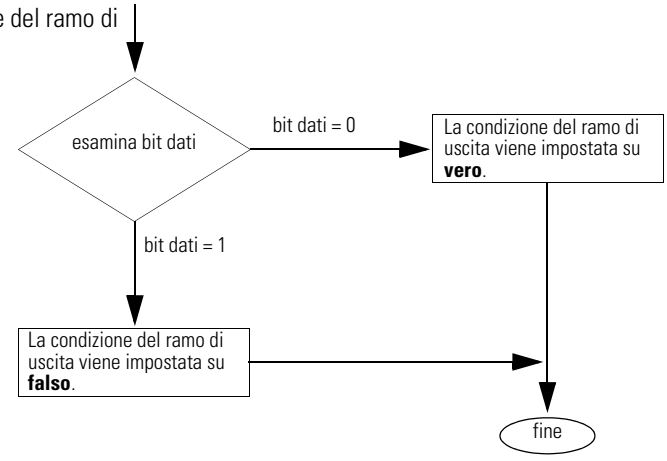
Consultare l'Appendice B, Attributi di un blocco funzione per informazioni sulla sintassi dei costrutti nel testo strutturato.

Descrizione: L'istruzione XIO esamina il bit di dati per verificare se è azzerato.

Indicatori di stato aritmetico: non influenzati

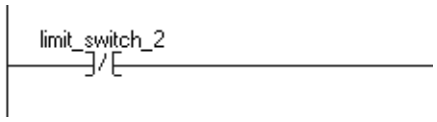
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio 1: Se *limit_switch_2* è azzerato, viene abilitata l'istruzione successiva (la condizione del ramo di uscita è vera).

Ladder

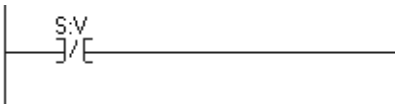


Testo strutturato

```
IF NOT limit_switch_2 THEN
    <istruzione>;
END_IF;
```

Esempio 2: Se *S:V* è azzerato (a indicare che non si è verificato alcun overflow), viene abilitata l'istruzione successiva (la condizione del ramo di uscita è vera).

Ladder



Testo strutturato

```
IF NOT S:V THEN
    <istruzione>;
END_IF;
```

Output Energize (OTE)

L'istruzione OTE imposta o azzera il bit di dati.

Operandi:

Ladder

Operando	Tipo	Formato	Descrizione
bit di dati	BOOL	tag	bit da impostare o azzerare

Testo strutturato

Il testo strutturato non ha un'istruzione OTE, ma è possibile ottenere gli stessi risultati usando un'assegnazione non ritentiva.

```
bit_dati [:=] espressione_booleana;
```

Consultare l'Appendice B, Attributi di un blocco funzione per informazioni sulla sintassi di assegnazioni ed espressioni nel testo strutturato.

Descrizione:

Quando l'istruzione OTE è abilitata, il controllore imposta il bit di dati. Quando l'istruzione OTE è disabilitata, il controllore azzera il bit di dati.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit di dati viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit di dati viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il bit di dati viene impostato. La condizione del ramo di uscita viene impostata su vero.
post-scansione	Il bit di dati viene azzerato. La condizione del ramo di uscita viene impostata su falso.

Esempio: Quando *switch* è impostato, l'istruzione OTE imposta (accende) *light_1*. Quando *switch* è azzerato, l'istruzione OTE azzera (spegne) *light_1*.

Ladder

Testo strutturato

```
light_1 [:=] switch;
```

Output Latch (OTL)

L'istruzione OTL imposta (aggancia) il bit di dati.

Operandi:

Ladder

Operando	Tipo	Formato	Descrizione
bit di dati	BOOL	tag	bit da impostare

Testo strutturato

Il testo strutturato non ha un'istruzione OTL, ma è possibile ottenere gli stessi risultati usando un costrutto IF...THEN e un'assegnazione.

```
IF espressione_booleana THEN
    bit_dati := 1;
END_IF;
```

Consultare l'Appendice B, Attributi di un blocco funzione per informazioni sulla sintassi di costrutti, espressioni e assegnazioni nel testo strutturato.

Descrizione: Quando è abilitata, l'istruzione OTL imposta il bit di dati. Il bit di dati rimane impostato finché non viene azzerato, di norma da un'istruzione OTU. Quando è disabilitata, l'istruzione OTL non modifica lo stato del bit di dati.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit di dati non viene modificato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit di dati non viene modificato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il bit di dati viene impostato. La condizione del ramo di uscita viene impostata su vero.
post-scansione	Il bit di dati non viene modificato. La condizione del ramo di uscita viene impostata su falso.

Esempio: Quando è abilitata, l'istruzione OTL imposta *light_2*. Il bit di dati rimane impostato finché non viene azzerato, di norma da un'istruzione OTU.

Ladder

Testo strutturato

```
IF espressione_booleana THEN
    light_2 := 1;
END_IF;
```

Output Unlatch (OTU)

L'istruzione OTU azzera (sgancia) il bit di dati.

Operandi:

Ladder

Operando	Tipo	Formato	Descrizione
bit di dati	BOOL	tag	bit da azzerare

Testo strutturato

Il testo strutturato non ha un'istruzione OTU, ma è possibile ottenere gli stessi risultati usando un costrutto IF...THEN e un'assegnazione.

```
IF espressione_booleana THEN
    bit_dati := 0;
END_IF;
```

Consultare l'Appendice B, Attributi di un blocco funzione per informazioni sulla sintassi di costrutti, espressioni e assegnazioni nel testo strutturato.

Descrizione: Quando è abilitata, l'istruzione OTU azzera il bit di dati. Quando è disabilitata, l'istruzione OTU non modifica lo stato del bit di dati.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit di dati non viene modificato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit di dati non viene modificato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il bit di dati viene azzerato. La condizione del ramo di uscita viene impostata su vero.
post-scansione	Il bit di dati non viene modificato. La condizione del ramo di uscita viene impostata su falso.

Esempio: Quando è abilitata, l'istruzione OTU azzera *light_2*.

Ladder

Testo strutturato

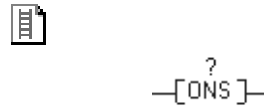
```
IF espressione_booleana THEN
    light_2 := 0;
END_IF;
```

One Shot (ONS)

L'istruzione ONS abilita o disabilita il resto del ramo, a seconda dello stato del bit di memoria.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
bit di memoria	BOOL	tag	bit di memoria interna memorizza la condizione del ramo di ingresso dell'ultima esecuzione dell'istruzione

Testo strutturato

Il testo strutturato non ha un'istruzione ONS, ma è possibile ottenere gli stessi risultati usando un costrutto IF...THEN.

```
IF espressione_booleana AND NOT bit_memoria THEN
    <istruzione>;
END_IF;
bit_memoria := espressione_booleana;
```

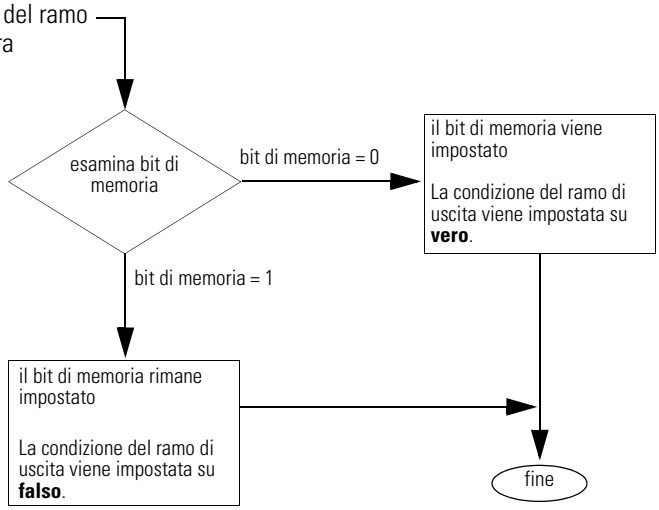
Consultare l'Appendice B, Attributi di un blocco funzione per informazioni sulla sintassi di costrutti, espressioni e assegnazioni nel testo strutturato.

Descrizione: Quando è abilitata e il bit di memoria è azzerato, l'istruzione ONS abilita il resto del ramo. Quando è disabilitata o il bit di memoria è impostato, l'istruzione ONS disabilita il resto del ramo.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit di memoria viene impostato per evitare un trigger non valido durante la prima scansione. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit di memoria viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	Il bit di memoria viene azzerato. La condizione del ramo di uscita viene impostata su falso.

Esempio: Di norma, si fa precedere l'istruzione ONS da un'istruzione di ingresso, dato che per un corretto funzionamento la scansione dell'istruzione ONS viene eseguita quando l'istruzione è abilitata e quando è disabilitata. Una volta che l'istruzione ONS è abilitata, la condizione del ramo di ingresso deve essere azzerata, oppure è necessario azzerare il bit di memoria per far sì che l'istruzione ONS venga abilitata di nuovo.

In qualunque scansione per la quale *limit_switch_1* viene azzerato o *storage_1* viene impostato, questo ramo non ha alcun effetto. In qualunque scansione per la quale *limit_switch_1* viene impostato e *storage_1 é* azzerato, l'istruzione ONS imposta *storage_1* e l'istruzione ADD aumenta *sum* di 1. Finché *limit_switch_1* rimane impostato, *sum* conserva lo stesso valore. *limit_switch_1* deve passare di nuovo da azzerato a impostato per far sì che *sum* torni ad essere aumentato.

Ladder

Testo strutturato

```
IF limit_switch_1 AND NOT storage_1 THEN
    sum := sum + 1;
END_IF;
storage_1 := limit_switch_1;
```

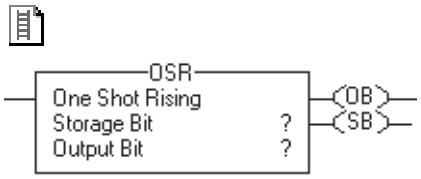
One Shot Rising (OSR)

L'istruzione OSR imposta o azzera il bit di uscita, a seconda dello stato del bit di memoria.

L'istruzione è disponibile in testo strutturato e blocco funzione come OSRI, vedere pagina 94.

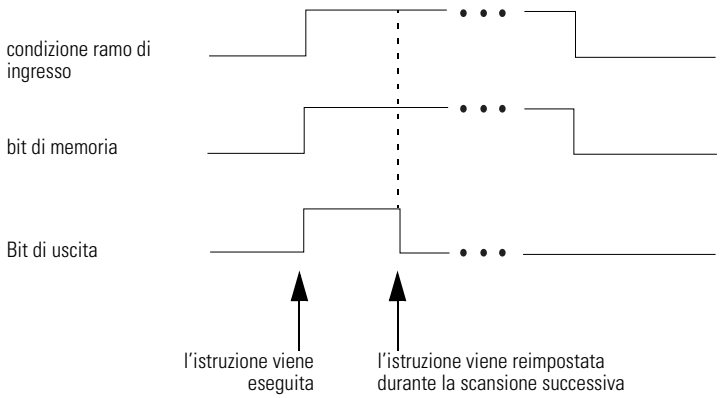
Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
bit di memoria	BOOL	tag	bit di memoria interna memorizza la condizione del ramo di ingresso dell'ultima esecuzione dell'istruzione
bit di uscita	BOOL	tag	bit da impostare

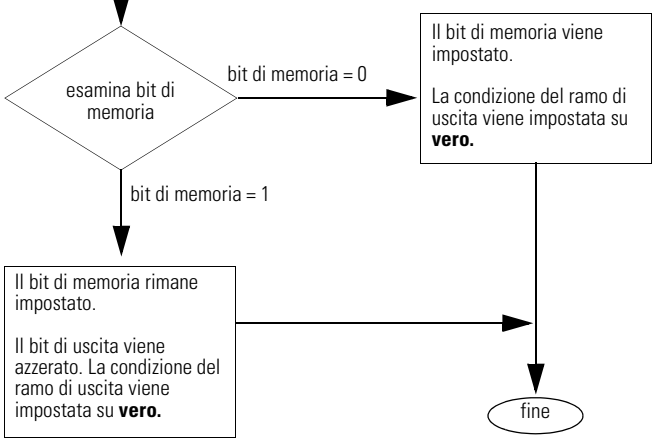
Descrizione: Quando è abilitata e il bit di memoria è azzerato, l'istruzione OSR imposta il bit di uscita. Quando è abilitata e il bit di memoria è impostato o quando è disabilitata, l'istruzione OSR azzera il bit di uscita


condizione ramo di ingresso
bit di memoria
Bit di uscita
l'istruzione viene eseguita
l'istruzione viene reimpostata durante la scansione successiva

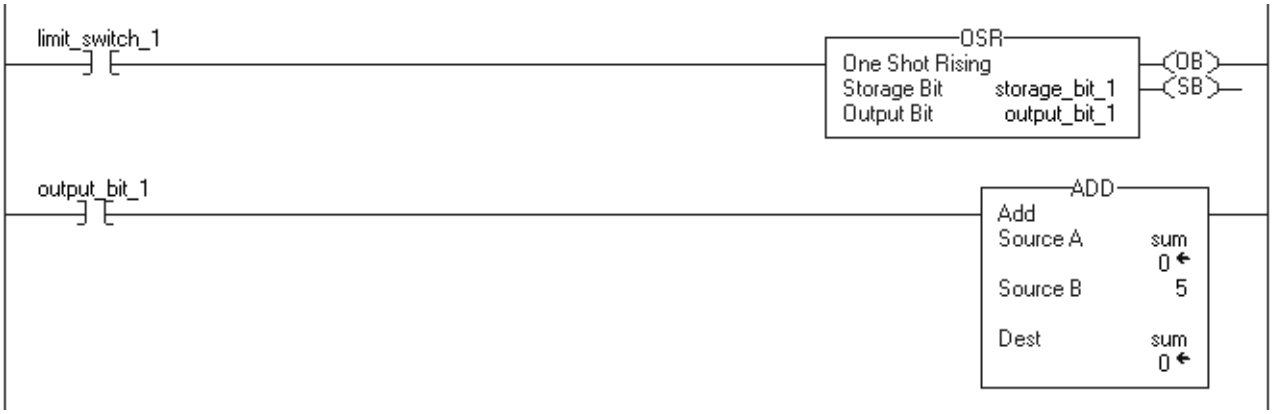
Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit di memoria viene impostato per evitare un trigger non valido durante la prima scansione. Il bit di uscita viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit di memoria viene azzerato. Il bit di uscita non viene modificato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	Il bit di memoria viene azzerato. Il bit di uscita non viene modificato. La condizione del ramo di uscita viene impostata su falso.

Esempio: Ogni volta che *limit_switch_1* passa da azzerato a impostato, l'istruzione OSR imposta *output_bit_1* e l'istruzione ADD aumenta *sum* di 5. Finché *limit_switch_1* rimane impostato, *sum* conserva lo stesso valore. *limit_switch_1* deve passare di nuovo da azzerato a impostato per far sì che *sum* torni ad essere aumentato. È possibile utilizzare *output_bit_1* su più rami per attivare altre operazioni



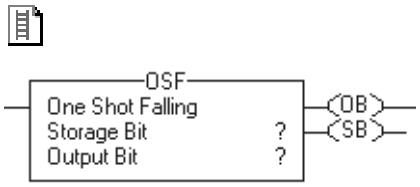
One Shot Falling (OSF)

L'istruzione OSF imposta o azzera il bit di uscita a seconda dello stato del bit di memoria.

L'istruzione è disponibile in testo strutturato e blocco funzione come OSFI, vedere pagina 97.

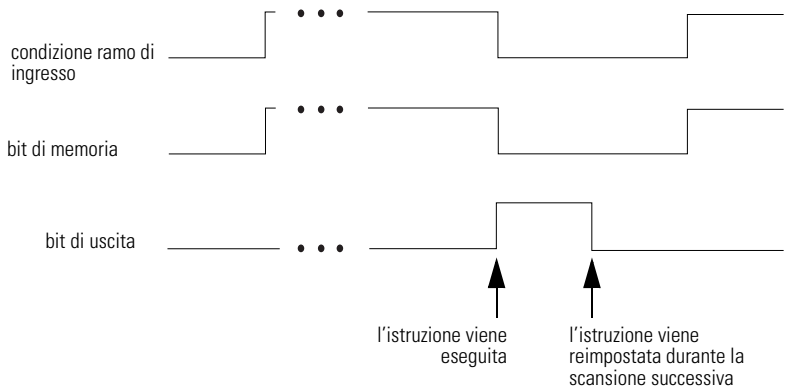
Operandi:

Operandi ladder



Operando	Tipo	Formato	Descrizione
bit di memoria	BOOL	tag	bit di memoria interna memorizza la condizione del ramo di ingresso dell'ultima esecuzione dell'istruzione
Bit di uscita	BOOL	tag	bit da impostare

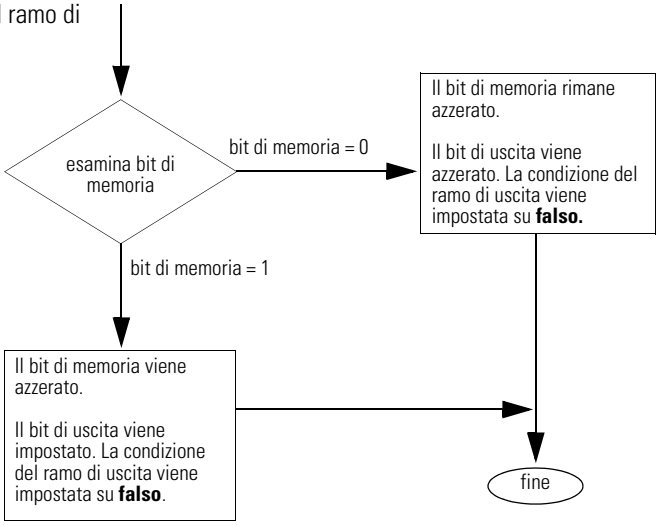
Descrizione: Quando è disabilitata e il bit di memoria è impostato, l'istruzione OSF imposta il bit di uscita. Quando è disabilitata e il bit di memoria è azzerato o quando è abilitata, l'istruzione OSF azzera il bit di uscita.


condizione ramo di ingresso
bit di memoria
bit di uscita
l'istruzione viene eseguita
l'istruzione viene reimpostata durante la scansione successiva

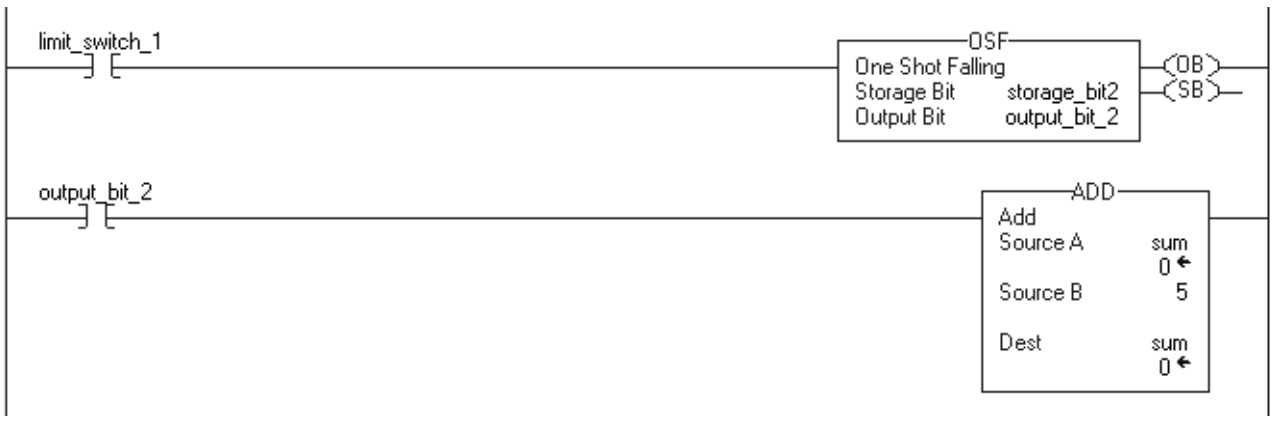
Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit di memoria viene azzerato per evitare un trigger non valido durante la prima scansione. Il bit di uscita viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	
se la condizione del ramo di ingresso è vera	Il bit di memoria viene impostato. Il bit di uscita viene azzerato. La condizione del ramo di uscita viene impostata su vero.
post-scansione	Vedere sopra, "Se la condizione del ramo di ingresso è falsa".

Esempio: Ogni volta che *limit_switch_1* passa da impostato ad azzerato, l'istruzione OSF imposta *output_bit_2* e l'istruzione ADD aumenta *sum* di 5. Finché *limit_switch_1* rimane azzerato, *sum* conserva lo stesso valore. *limit_switch_1* deve passare di nuovo da impostato ad azzerato per far sì che *sum* torni ad essere aumentato. È possibile utilizzare *output_bit_2* su più rami per attivare altre operazioni



One Shot Rising with Input (OSRI)

L'istruzione OSRI imposta il bit di uscita per un ciclo di esecuzione quando il bit di ingresso passa da azzerato a impostato.

Questa istruzione è disponibile in ladder come OSR, vedere pagina 89.

Operandi:

```
OSRI(OSRI_tag);
```

Testo strutturato

Operando	Tipo	Formato	Descrizione
tag OSRI	FBD_ONESHOT	struttura	struttura OSRI

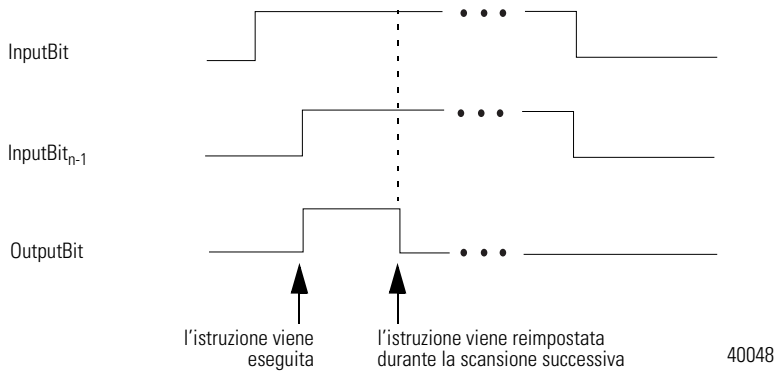
Blocco funzione

Operando	Tipo	Formato	Descrizione
tag OSRI	FBD_ONESHOT	struttura	struttura OSRI

Struttura FBD_ONESHOT

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato:** Nessun effetto. L'istruzione viene eseguita.
InputBit	BOOL	Bit di ingresso. Equivale alla condizione del ramo dell'istruzione ladder OSR. Il valore di default è azzerato.
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
OutputBit	BOOL	Bit di uscita

Descrizione: Quando InputBit è impostato e $InputBit_{n-1}$ è azzerato, l'istruzione OSRI imposta OutputBit. Quando $InputBit_{n-1}$ è impostato o quando InputBit è azzerato, l'istruzione OSRI azzera OutputBit.



Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

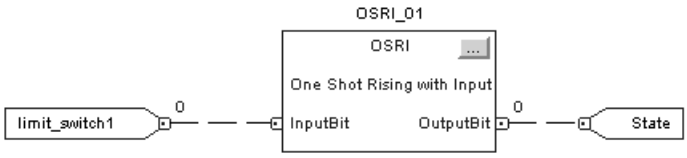
Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Nessuna.	Nessuna.
prima scansione dell'istruzione	$InputBit_{n-1}$ viene impostato.	$InputBit_{n-1}$ viene impostato.
durante la prima esecuzione dell'istruzione	$InputBit_{n-1}$ viene impostato.	$InputBit_{n-1}$ viene impostato.
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.	na
EnableIn viene impostato	Nella transizione da azzerato a impostato di InputBit, l'istruzione imposta $InputBit_{n-1}$. L'istruzione viene eseguita. EnableOut viene impostato.	Nella transizione da azzerato a impostato di InputBit, l'istruzione imposta $InputBit_{n-1}$. EnableIn viene sempre impostato. L'istruzione viene eseguita.
post-scansione	Nessuna.	Nessuna.

Esempio: Quando *limit_switch1* passa da azzerato a impostato, l'istruzione OSRI imposta OutputBit per una scansione.

Testo strutturato

```
OSRI_01.InputBit := limit_switch1;

OSRI(OSRI_01);

state := OSRI_01.OutputBit;
```

Blocco funzione



One Shot Falling with Input (OSFI)

L'istruzione OSFI imposta OutputBit per un ciclo di esecuzione quando InputBit passa da impostato ad azzerato.

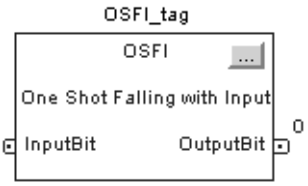
Questa istruzione è disponibile in ladder come OSF, vedere pagina 92.

Operandi:

```
OSFI(OSFI_tag);
```

Testo strutturato

Operando	Tipo	Formato	Descrizione
tag OSFI	FBD_ONESHOT	struttura	struttura OSFI



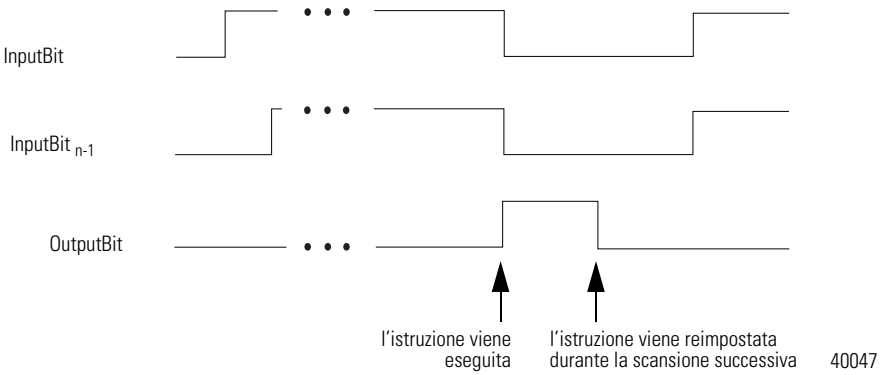
Blocco funzione

Operando	Tipo	Formato	Descrizione
tag OSFI	FBD_ONESHOT	struttura	struttura OSFI

Struttura FBD_ONESHOT

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato:** Nessun effetto. L'istruzione viene eseguita.
InputBit	BOOL	Bit di ingresso. Equivale alla condizione del ramo dell'istruzione ladder OSF Il valore di default è azzerato.
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
OutputBit	BOOL	Bit di uscita

Descrizione: Quando InputBit è azzerato e $InputBit_{n-1}$ è impostato, l'istruzione OSFI imposta OutputBit. Quando $InputBit_{n-1}$ è azzerato o quando InputBit è impostato, l'istruzione OSFI azzera OutputBit.



Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

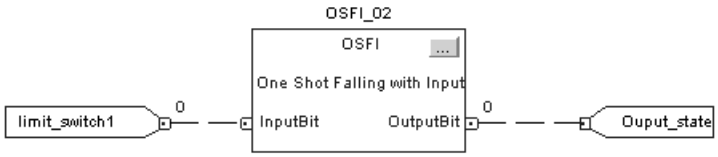
Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Nessuna.	Nessuna.
prima scansione dell'istruzione	$InputBit_{n-1}$ viene azzerato.	$InputBit_{n-1}$ viene azzerato.
durante la prima esecuzione dell'istruzione	$InputBit_{n-1}$ viene azzerato.	$InputBit_{n-1}$ viene azzerato.
EnableIn é azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.	na
EnableIn é impostato	Nella transizione da azzerato a impostato di InputBit, l'istruzione azzera $InputBit_{n-1}$. L'istruzione viene eseguita. EnableOut viene impostato.	Nella transizione da azzerato a impostato di InputBit, l'istruzione azzera $InputBit_{n-1}$. EnableIn viene sempre impostato. L'istruzione viene eseguita.
post-scansione	Nessuna.	Nessuna.

Esempio: Quando *limit_switch1* passa da impostato ad azzerato, l'istruzione OSFI imposta OutputBit per una scansione.

Testo strutturato

```
OSFI_01.InputBit := limit_switch1;

OSFI(OSFI_01);

Output_state := OSFI_01.OutputBit;
```

Blocco funzione



Note:

Istruzioni di timer e contatori

(TON, TOF, RTO, TONR, TOFR, RTOR, CTU, CTD, CTUD, RES)

Introduzione

I timer e i contatori controllano le operazioni in base al tempo o al numero di eventi.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
calcolare per quanto tempo un timer è abilitato	TON	ladder	102
calcolare per quanto tempo un timer è disabilitato	TOF	ladder	106
accumulare il tempo	RTO	ladder	110
calcolare per quanto tempo un timer è abilitato con reset incorporato nel blocco funzione	TONR	testo strutturato blocco funzione	114
calcolare per quanto tempo un timer è disabilitato con reset incorporato nel blocco funzione	TOFR	testo strutturato blocco funzione	118
accumulare il tempo con reset incorporato nel blocco funzione	RTOR	testo strutturato blocco funzione	122
contare in modo incrementale	CTU	ladder	126
contare in modo decrementale	CTD	ladder	130
contare in modo incrementale e decrementale nel blocco funzione	CTUD	testo strutturato blocco funzione	134
azzerare un timer o un contatore	RES	ladder	139

La base tempo di tutti i timer è 1 msec.

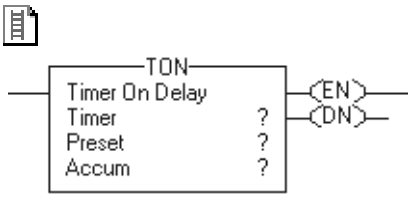
Timer On Delay (TON)

L'istruzione TON è un timer non ritentivo che calcola il tempo quando l'istruzione è abilitata (la condizione del ramo di ingresso è vera).

L'istruzione è disponibile in testo strutturato e blocco funzione come TONR, vedere pagina 114.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Timer	TIMER	tag	struttura TIMER
Preset	DINT	immediato	quanto ritardare (tempo accumulato)
Accum	DINT	immediato	msec. totali calcolati dal timer il valore iniziale generalmente è 0

Struttura TIMER

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione TON è abilitata.
.TT	BOOL	Il bit di temporizzazione indica che è in corso un'operazione di temporizzazione
.DN	BOOL	Il bit di fine viene impostato quando .ACC ≥ .PRE.
.PRE	DINT	Il valore preimpostato specifica il valore (unità di 1 msec) che il valore accumulato deve raggiungere prima che l'istruzione imposti il bit .DN.
.ACC	DINT	Il valore accumulato specifica il numero di millisecondi trascorsi dall'abilitazione dell'istruzione TON.

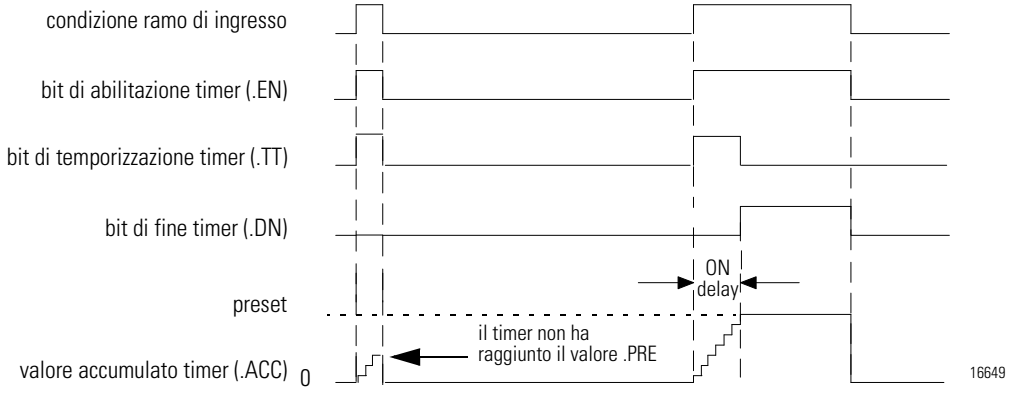
Descrizione:

L'istruzione TON calcola il tempo finché:

- l'istruzione TON non viene disabilitata
- .ACC ≥ .PRE

La base tempo è sempre 1 msec. Ad esempio, per un timer da 2 secondi, immettere 2000 per il valore .PRE.

Quando l'istruzione TON è disabilitata, il valore .ACC viene azzerato.



Un timer funziona sottraendo il tempo dell'ultima scansione dal tempo corrente:

ACC = ACC + (*tempo_corrente* – *tempo_ultima_scansione*)

Una volta aggiornato il valore ACC, il timer imposta *tempo_ultima_scansione* = *tempo_corrente*. In questo modo il timer è pronto per la scansione successiva.

IMPORTANTE

Accertarsi di scansionare il timer almeno ogni 69 minuti quando è in funzione. Diversamente, il valore ACC non sarà corretto.

Il valore *tempo_ultima_scansione* ha un intervallo massimo di 69 minuti. Il calcolo del timer ricomincia daccapo se non viene effettuata la scansione entro i 69 minuti. In questo caso, il valore ACC non sarà corretto.

Mentre un timer è in esecuzione, eseguire la scansione entro 69 minuti se lo si colloca all'interno di:

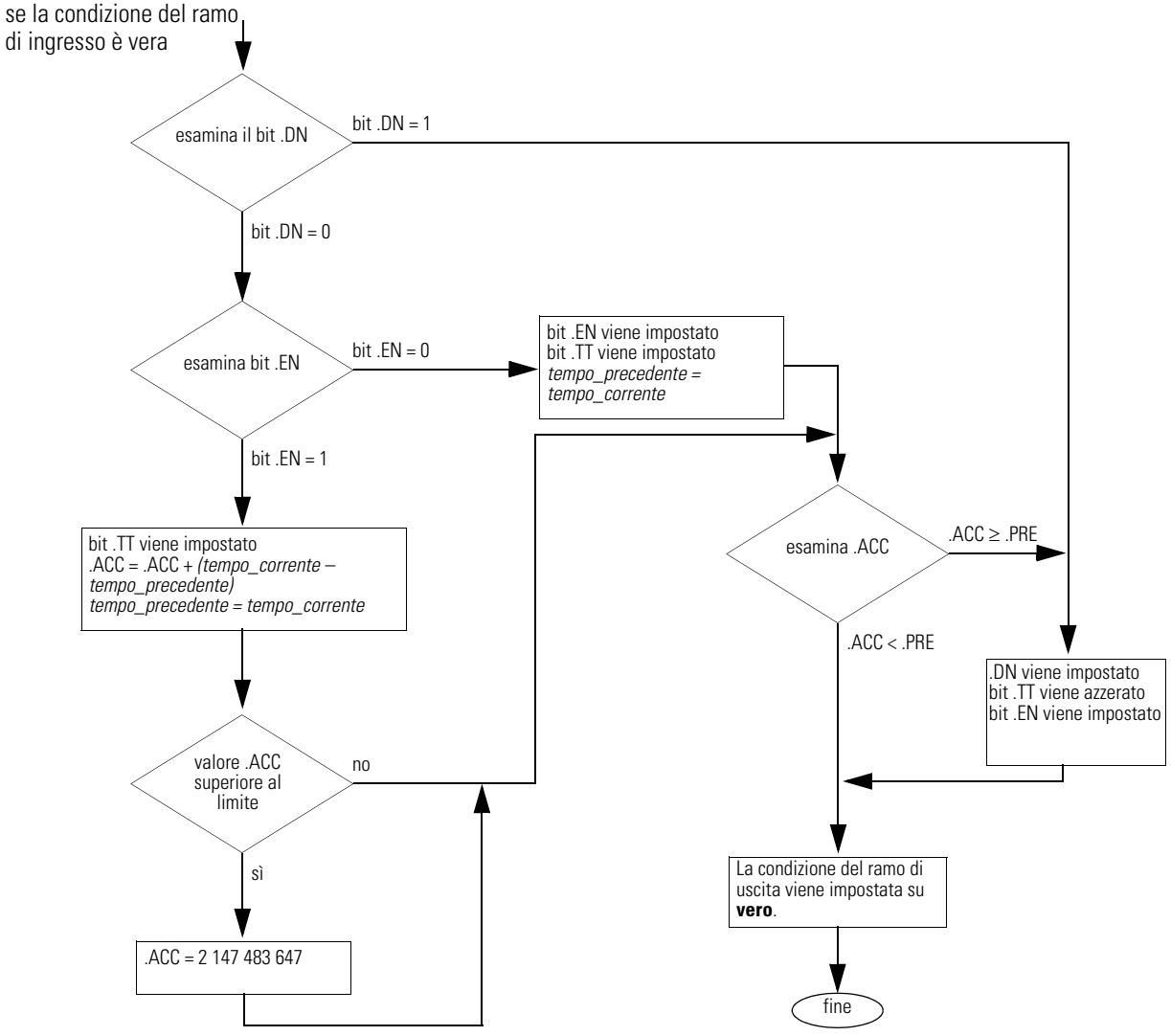
- una subroutine
- una sezione di codice compresa tra le istruzioni JMP e LBL
- un diagramma funzionale sequenziale (SFC)
- un evento o task periodico
- una routine di stato di una fase

Indicatori di stato aritmetico: non influenzati

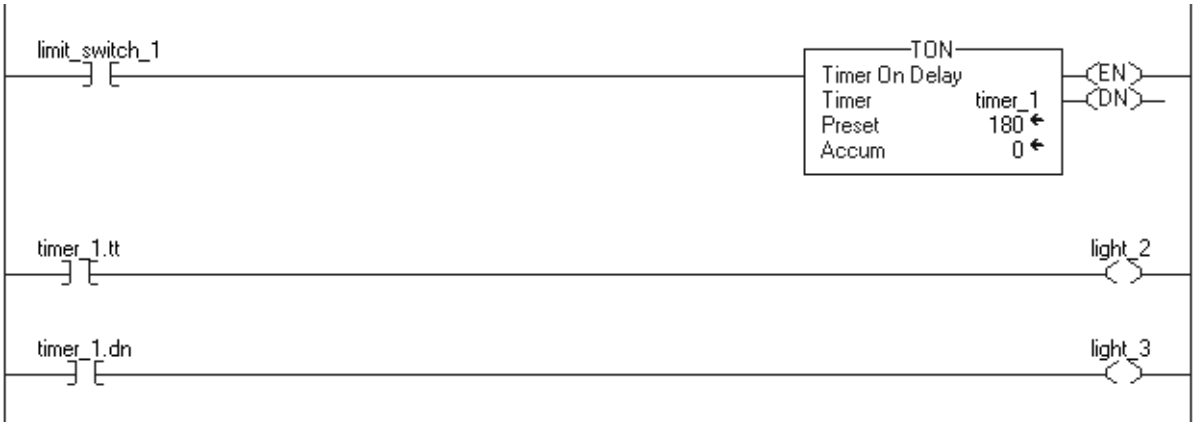
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
.PRE < 0	4	34
.ACC < 0	4	34

Esecuzione:

Condizione	Azione ladder
durante la prescansione	I bit .EN, .TT e .DN vengono azzerati. Il valore .ACC viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	I bit .EN, .TT e .DN vengono azzerati. Il valore .ACC viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Quando *limit_switch_1* è impostato, *light_2* è accesa per 180 msec (*timer_1* è in fase di conteggio). Quando *timer_1.acc* raggiunge 180, *light_2* si spegne e *light_3* si accende. *Light_3* rimane accesa finché l'istruzione TON non viene disabilitata. Se *limit_switch_1* viene azzerato mentre *timer_1* è in fase di conteggio, *light_2* si spegne.



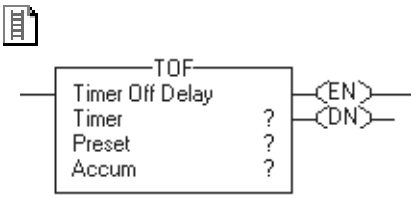
Timer Off Delay (TOF)

L'istruzione TOF è un timer non ritentivo che calcola il tempo quando l'istruzione è abilitata (la condizione del ramo di ingresso è falsa).

Questa istruzione è disponibile in testo strutturato e blocco funzione come TOFR, vedere pagina 118.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Timer	TIMER	tag	struttura TIMER
Preset	DINT	immediato	quanto ritardare (tempo accumulato)
Accum	DINT	immediato	msec. totali calcolati dal timer il valore iniziale generalmente è 0

Struttura TIMER

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione TOF è abilitata.
.TT	BOOL	Il bit di temporizzazione indica che è in corso un'operazione di temporizzazione
.DN	BOOL	Il bit di fine viene azzerato quando .ACC ≥ .PRE.
.PRE	DINT	Il valore preimpostato specifica il valore (unità di 1 msec) che il valore accumulato deve raggiungere prima che l'istruzione azzeri il bit .DN.
.ACC	DINT	Il valore accumulato specifica il numero di millisecondi trascorsi dall'abilitazione dell'istruzione TOF.

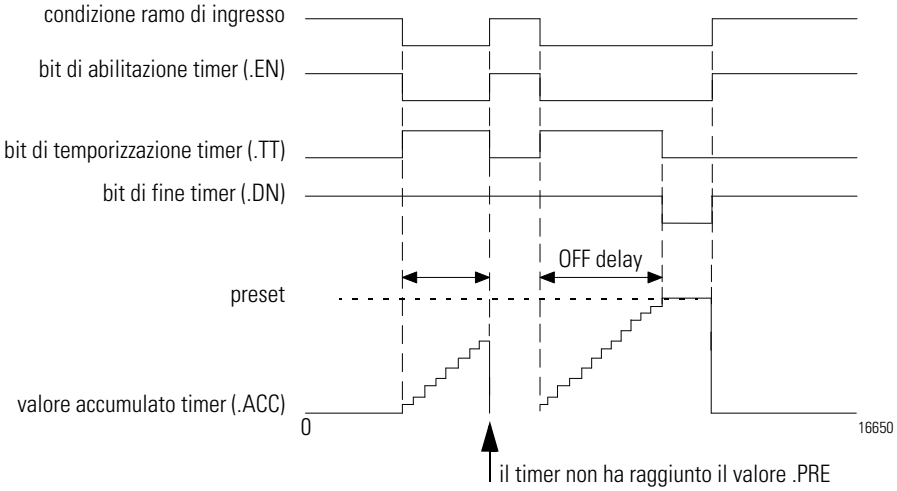
Descrizione:

L'istruzione TOF calcola il tempo finché:

- l'istruzione TOF non viene disabilitata
- .ACC ≥ .PRE

La base tempo è sempre 1 msec. Ad esempio, per un timer da 2 secondi, immettere 2000 per il valore .PRE.

Quando l'istruzione TOF è disabilitata, il valore .ACC viene azzerato.



Un timer funziona sottraendo il tempo dell'ultima scansione dal tempo corrente:

ACC = ACC + (*tempo_corrente* – *tempo_ultima_scansione*)

Una volta aggiornato il valore ACC, il timer imposta *tempo_ultima_scansione* = *tempo_corrente*. In questo modo il timer è pronto per la scansione successiva.

IMPORTANTE

Accertarsi di scansionare il timer almeno ogni 69 minuti quando è in funzione. Diversamente, il valore ACC non sarà corretto.

Il valore *tempo_ultima_scansione* ha un intervallo massimo di 69 minuti. Il calcolo del timer ricomincia daccapo se non viene effettuata la scansione entro i 69 minuti. In questo caso, il valore ACC non sarà corretto.

Mentre un timer è in esecuzione, eseguire la scansione entro 69 minuti se lo si colloca all'interno di:

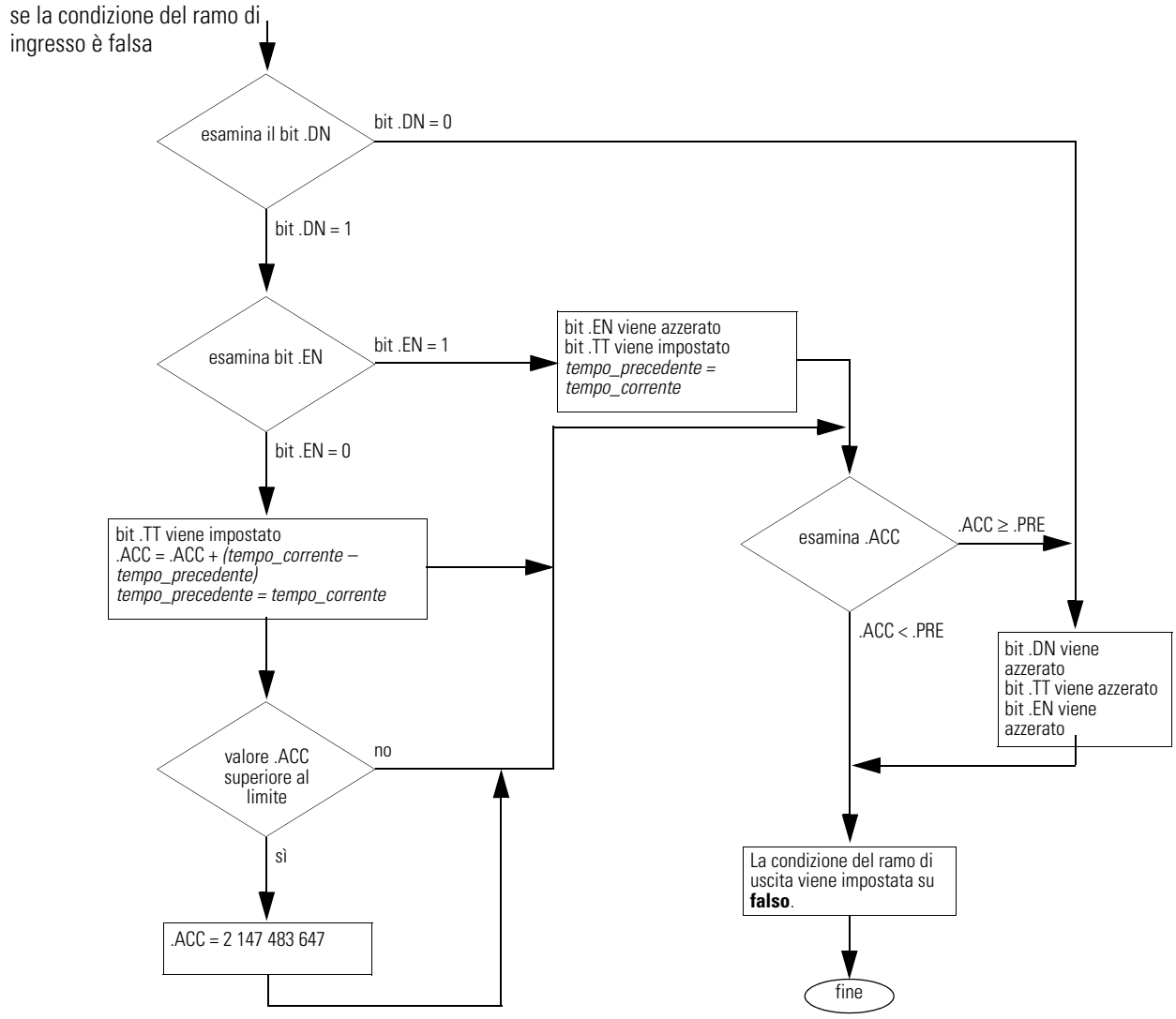
- subroutine
- una sezione di codice compresa tra le istruzioni JMP e LBL
- un diagramma funzionale sequenziale (SFC)
- un evento o task periodico
- una routine di stato di una fase

Indicatori di stato aritmetico: non influenzati

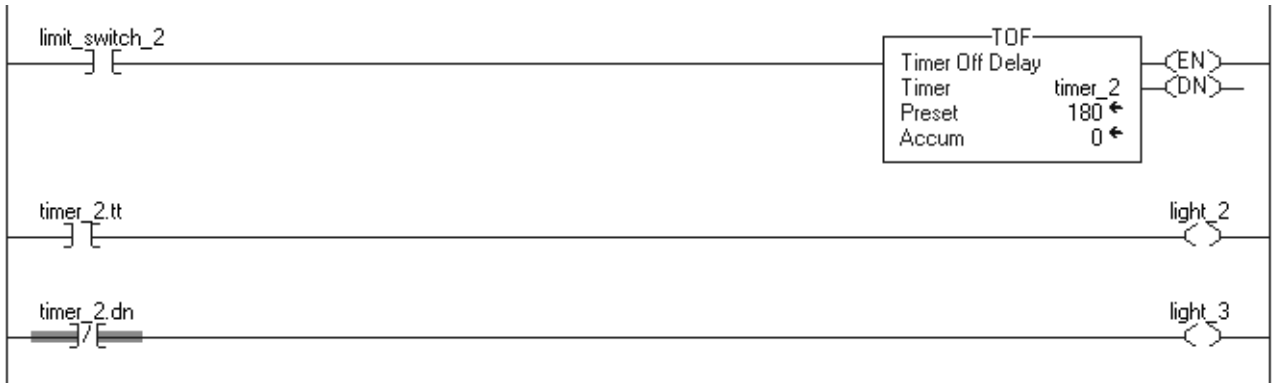
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
.PRE < 0	4	34
.ACC < 0	4	34

Esecuzione:

Condizione	Azione ladder
durante la prescansione	I bit .EN, .TT e .DN vengono azzerati. Il valore .ACC viene impostato uguale al valore .PRE. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	
se la condizione del ramo di ingresso è vera	I bit .EN, .TT e .DN vengono impostati. Il valore .ACC viene azzerato. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Quando *limit_switch_2* viene azzerato, *light_2* è accesa per 180 msec (*timer_2* è in fase di conteggio). Quando *timer_2.acc* raggiunge 180, *light_2* si spegne e *light_3* si accende. *Light_3* rimane accesa finché l'istruzione TOF non viene abilitata. Se *limit_switch_2* viene impostato mentre *timer_2* è in fase di conteggio, *light_2* si spegne.



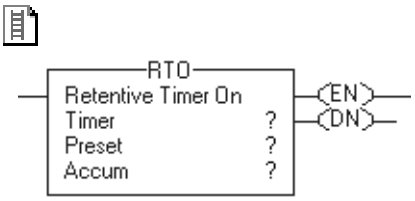
Retentive Timer On (RTO)

L'istruzione RTO è un timer ritentivo che calcola il tempo quando l'istruzione è abilitata.

L'istruzione è disponibile in testo strutturato e blocco funzione come RTOR, vedere pagina 122.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Timer	TIMER	tag	struttura TIMER
Preset	DINT	immediato	quanto ritardare (tempo accumulato)
Accum	DINT	immediato	numero di msec calcolati dal timer il valore iniziale generalmente è 0

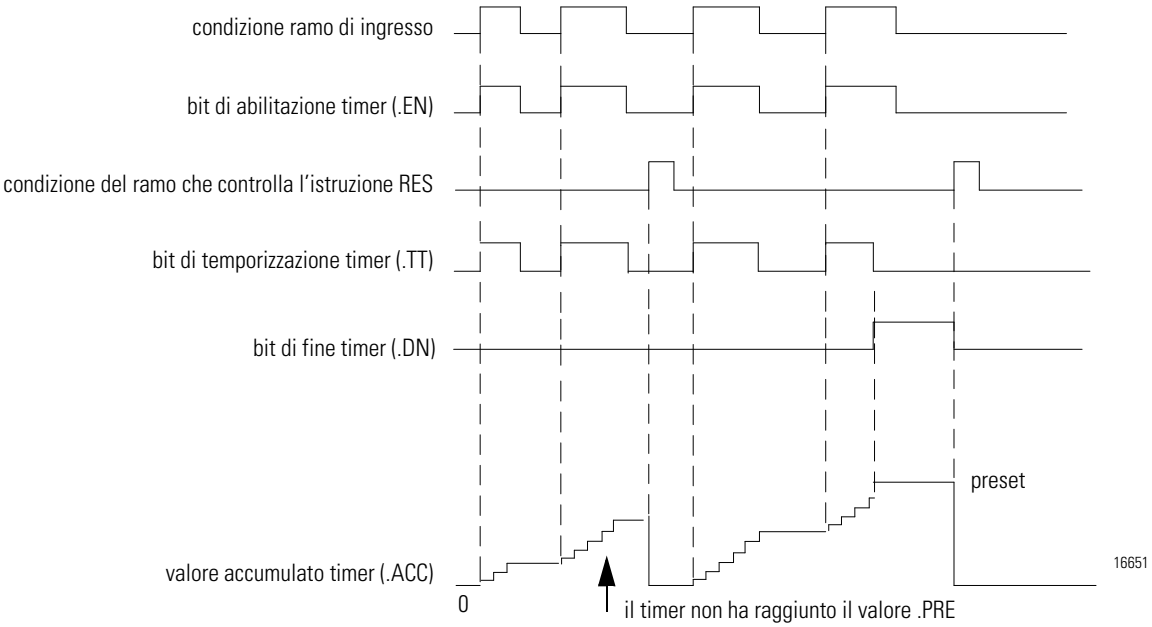
Struttura TIMER

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione RTO è abilitata.
.TT	BOOL	Il bit di temporizzazione indica che è in corso un'operazione di temporizzazione
.DN	BOOL	Il bit di fine indica che .ACC ≥ .PRE.
.PRE	DINT	Il valore preimpostato specifica il valore (unità di 1 msec) che il valore accumulato deve raggiungere prima che l'istruzione imposti il bit .DN.
.ACC	DINT	Il valore accumulato specifica il numero di millisecondi trascorsi dall'abilitazione dell'istruzione RTO.

Descrizione:

L'istruzione RTO calcola il tempo finché non viene disabilitata. Quando è disabilitata, l'istruzione RTO conserva il proprio valore .ACC. È necessario azzerare il valore .ACC, di norma con un'istruzione RES che faccia riferimento alla stessa struttura TIMER.

La base tempo è sempre 1 msec. Ad esempio, per un timer da 2 secondi, immettere 2000 per il valore .PRE.


condizione ramo di ingresso
bit di abilitazione timer (.EN)
condizione del ramo che controlla l'istruzione RES
bit di temporizzazione timer (.TT)
bit di fine timer (.DN)
preset
valore accumulato timer (.ACC)
16651
0
il timer non ha raggiunto il valore .PRE

Un timer funziona sottraendo il tempo dell'ultima scansione dal tempo corrente:

ACC = ACC + (*tempo_corrente* – *tempo_ultima_scansione*)

Una volta aggiornato il valore ACC, il timer imposta *tempo_ultima_scansione* = *tempo_corrente*. In questo modo il timer è pronto per la scansione successiva.

IMPORTANTE

Accertarsi di scansionare il timer almeno ogni 69 minuti quando è in funzione. Diversamente, il valore ACC non sarà corretto.

Il valore *tempo_ultima_scansione* ha un intervallo massimo di 69 minuti. Il calcolo del timer ricomincia daccapo se non viene effettuata la scansione entro i 69 minuti. In questo caso, il valore ACC non sarà corretto.

Mentre un timer è in esecuzione, eseguire la scansione entro 69 minuti se lo si colloca all'interno di:

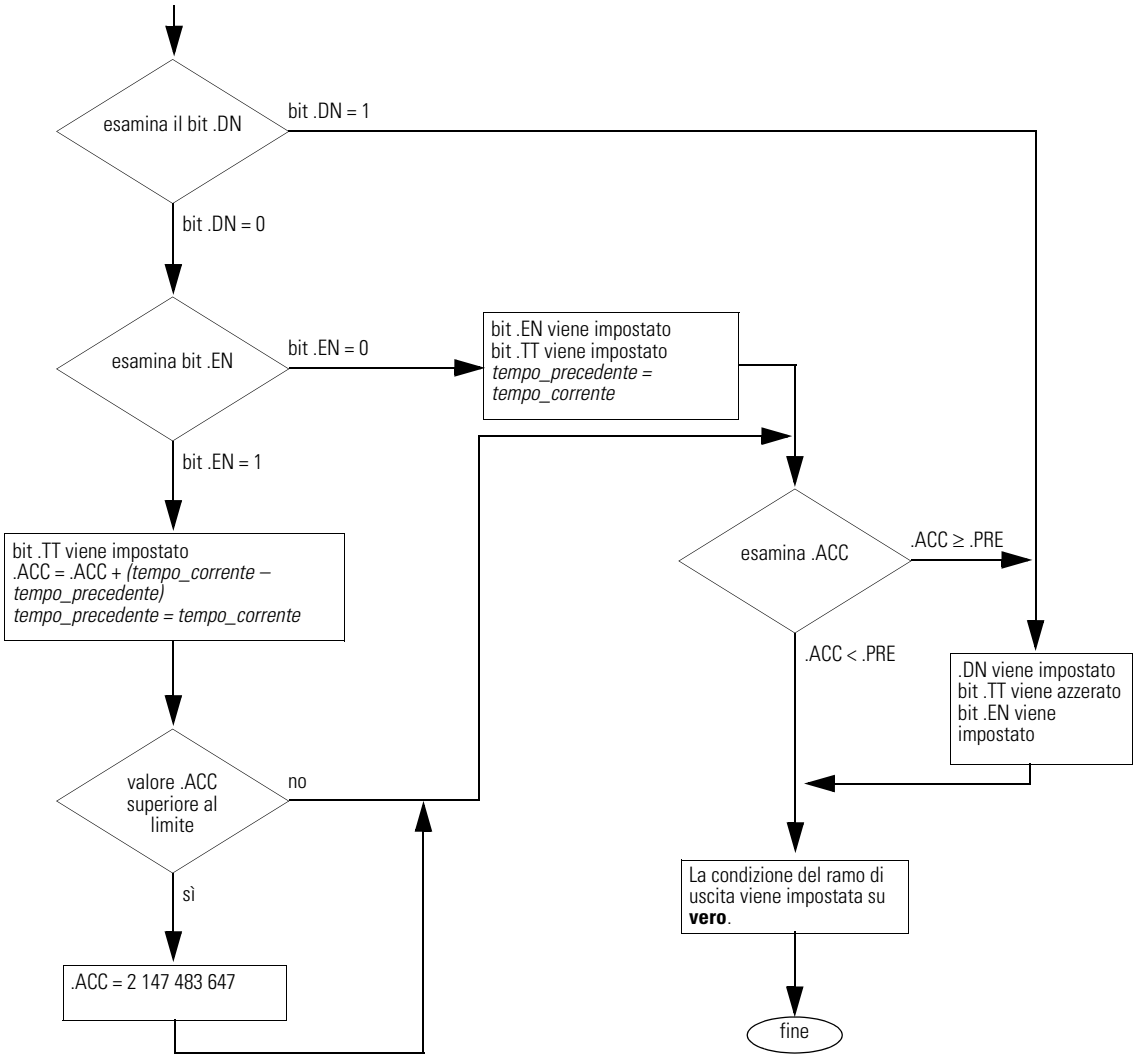
- subroutine
- una sezione di codice compresa tra le istruzioni JMP e LBL
- un diagramma funzionale sequenziale (SFC)
- un evento o task periodico
- una routine di stato di una fase

Indicatori di stato aritmetico: non influenzati

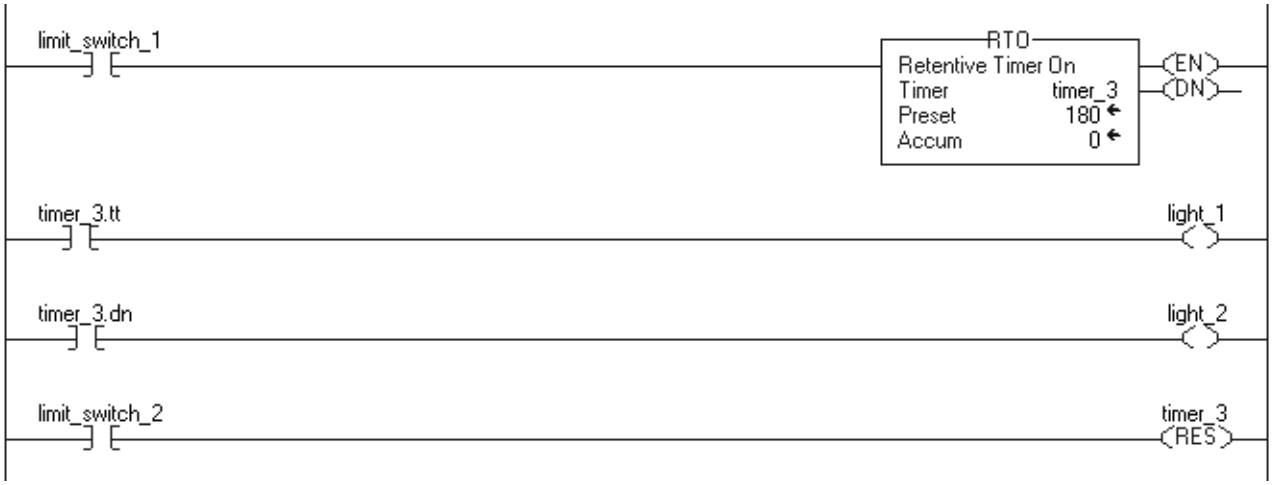
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
.PRE < 0	4	34
.ACC < 0	4	34

Esecuzione:

Condizione	Azione ladder
durante la prescansione	I bit .EN, .TT e .DN vengono azzerati. Il valore .ACC non viene modificato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	I bit .EN e .TT vengono azzerati. Il bit .DN non viene modificato. Il valore .ACC non viene modificato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Quando *limit_switch_1* è impostato, *light_1* è accesa per 180 msec (*timer_2* è in fase di conteggio). Quando *timer_3.acc* raggiunge 180, *light_1* si spegne e *light_2* si accende. *Light_2* rimane accesa finché *timer_3* non viene reimpostato. Se *limit_switch_2* viene azzerato mentre *timer_3* è in fase di conteggio, *light_1* rimane accesa. Quando *limit_switch_2* viene impostato, l'istruzione RES reimposta *timer_3* (azzera i bit di stato e il valore .ACC).



Timer On Delay with Reset (TONR)

L'istruzione TONR è un timer non ritentivo che conteggia il tempo quando TimerEnable è impostato.

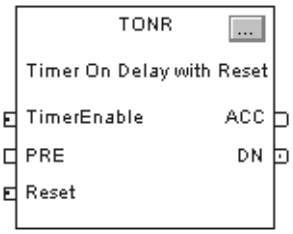
Questa istruzione è disponibile in ladder sotto forma di due istruzioni separate: TON (vedere pagina 102) e RES (vedere pagina 139).

Operandi:

```
TONR(TONR_tag);
```

Testo strutturato

Variabile	Tipo	Formato	Descrizione
tag TONR	FBD_TIMER	struttura	struttura TONR



Blocco funzione

Operando	Tipo	Formato	Descrizione
tag TONR	FBD_TIMER	struttura	struttura TONR

Struttura FBD_TIMER

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato:** nessun effetto. L'istruzione viene eseguita.
TimerEnable	BOOL	Se è impostato, il timer viene eseguito e conteggia il tempo. Il valore di default è azzerato.
PRE	DINT	Valore preimpostato del timer. Si tratta del valore, in unità di 1msec, che ACC deve raggiungere prima che il rilevamento del tempo venga terminato. Se non è valido, l'istruzione imposta il bit adeguato in Status e il timer non viene eseguito. Valido = da 0 al massimo intero positivo
reset	BOOL	Richiesta di reimpostazione del timer. Quando è impostato, il timer viene reimpostato. Il valore di default è azzerato.

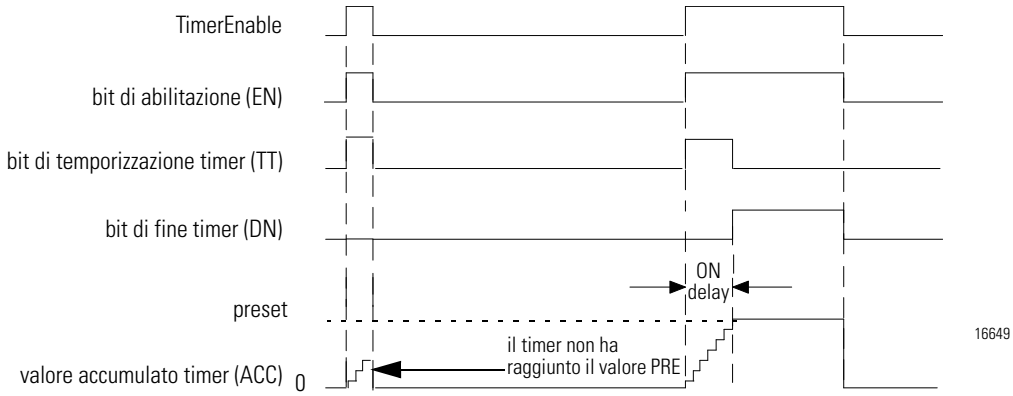
Parametro di uscita	Tipo di dati	Descrizione
EnableOut	BOOL	L'istruzione produce un risultato valido.
ACC	BOOL	Tempo accumulato in millisecondi.
EN	BOOL	Uscita di abilitazione timer. Indica che l'istruzione timer è abilitata.
TT	BOOL	Uscita di esecuzione timer. Quando è impostato, è in corso un'operazione di rilevamento del tempo.
DN	BOOL	Uscita di fine rilevazione tempo. Indica quando il tempo accumulato è maggiore di o uguale al valore preimpostato.

Parametro di ingresso	Tipo di dati	Descrizione
Status	DINT	Stato del blocco funzione.
InstructFault (Status.0)	BOOL	L'istruzione ha rilevato uno dei seguenti errori di esecuzione. Non si tratta di un errore minore o grave del controllore. Verificare gli altri bit di stato per stabilire cosa è accaduto.
PresetInv (Status.1)	BOOL	Il valore preimpostato non è valido.

Descrizione: L'istruzione TONR calcola il tempo finché:

- l'istruzione TONR non viene disabilitata
- ACC ≥ PRE

La base tempo è sempre 1 msec. Ad esempio, per un timer da 2 secondi, immettere 2000 per il valore .PRE.



Impostare il parametro di ingresso Reset per reimpostare l'istruzione. Se TimerEnable è impostato quando Reset viene impostato, l'istruzione TONR inizia a rilevare di nuovo il tempo quando Reset viene azzerato.

Un timer funziona sottraendo il tempo dell'ultima scansione dal tempo corrente:

ACC = ACC + (*tempo_corrente* – *tempo_ultima_scansione*)

Una volta aggiornato il valore ACC, il timer imposta *tempo_ultima_scansione* = *tempo_corrente*. In questo modo il timer è pronto per la scansione successiva.

IMPORTANTE

Accertarsi di scansionare il timer almeno ogni 69 minuti quando è in funzione. Diversamente, il valore ACC non sarà corretto.

Il valore *tempo_ultima_scansione* ha un intervallo massimo di 69 minuti. Il calcolo del timer ricomincia daccapo se non viene effettuata la scansione entro i 69 minuti. In questo caso, il valore ACC non sarà corretto.

Mentre un timer è in esecuzione, eseguire la scansione entro 69 minuti se lo si colloca all'interno di:

- subroutine
- una sezione di codice compresa tra le istruzioni JMP e LBL
- un diagramma funzionale sequenziale (SFC)
- un evento o task periodico
- una routine di stato di una fase

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

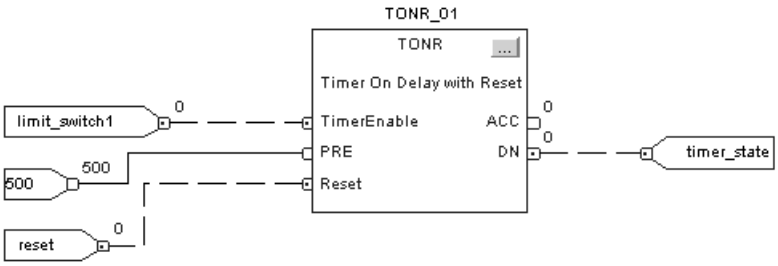
Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Nessuna.	Nessuna.
prima scansione dell'istruzione	EN, TT e DN vengono azzerati. Il valore ACC viene impostato a 0.	EN, TT e DN vengono azzerati. Il valore ACC viene impostato a 0.
durante la prima esecuzione dell'istruzione	EN, TT e DN vengono azzerati. Il valore ACC viene impostato a 0.	EN, TT e DN vengono azzerati. Il valore ACC viene impostato a 0.
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.	na
EnableIn viene impostato	Quando EnableIn passa da azzerato a impostato, l'istruzione viene inizializzata come descritto per la prima scansione dell'istruzione. L'istruzione viene eseguita. EnableOut viene impostato.	EnableIn viene sempre impostato. L'istruzione viene eseguita.
reset	Quando il parametro di ingresso Reset viene impostato, l'istruzione azzera EN, TT e DN e imposta ACC = zero.	Quando il parametro di ingresso Reset viene impostato, l'istruzione azzera EN, TT e DN e imposta ACC = zero.
post-scansione	Nessuna.	Nessuna.

Esempio: A ogni scansione in cui *limit_switch1* è impostato, l'istruzione TONR incrementa il valore ACC con il tempo trascorso finché il valore ACC non raggiunge il valore PRE. Quando ACC ≥ PRE, il parametro DN e *timer_state* vengono impostati.

Testo strutturato

```
TONR_01.Preset := 500;
TONR_01.Reset : = reset;
TONR_O1.TimerEnable := limit_switch1;

TONR(TONR_01);

timer_state := TONR_01.DN;
```

Esempio di blocco funzione



Timer Off Delay with Reset (TOFR)

L'istruzione TOFR è un timer non ritentivo che calcola il tempo quando TimerEnable è azzerato.

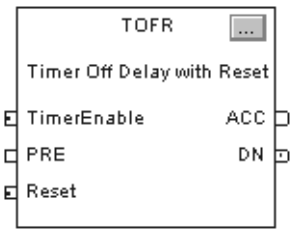
Questa istruzione è disponibile in ladder sotto forma di due istruzioni separate: TOF (vedere pagina 106) e RES (vedere pagina 139).

Operandi:

```
TOFR(TOFR_tag);
```

Testo strutturato

Variabile	Tipo	Formato	Descrizione
tag TOFR	FBD_TIMER	struttura	struttura TOFR



Operandi blocco funzione

Operando	Tipo	Formato	Descrizione
tag TOFR	FBD_TIMER	struttura	struttura TOFR

Struttura FBD_TIMER

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato:** nessun effetto. L'istruzione viene eseguita.
TimerEnable	BOOL	Se azzerato, il timer viene eseguito e calcola il tempo. Il valore di default è azzerato.
PRE	DINT	Valore preimpostato del timer. Si tratta del valore, in unità di 1 msec, che ACC deve raggiungere prima che il rilevamento del tempo venga terminato. Se non è valido, l'istruzione imposta il bit adeguato in Status e il timer non viene eseguito. Valido = da 0 al massimo intero positivo
reset	BOOL	Richiesta di reimpostazione del timer. Quando è impostato, il timer viene reimpostato. Il valore di default è azzerato.

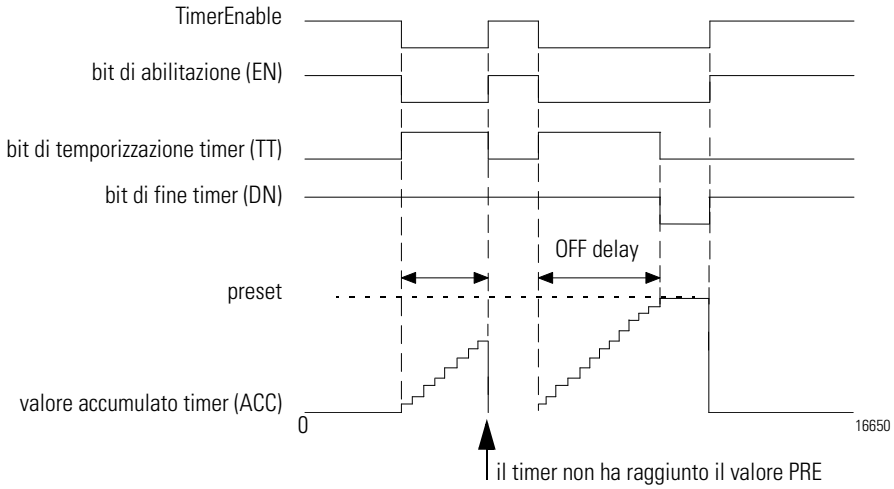
Parametro di uscita	Tipo di dati	Descrizione
EnableOut	BOOL	L'istruzione produce un risultato valido.
ACC	BOOL	Tempo accumulato in millisecondi.
EN	BOOL	Uscita di abilitazione timer. Indica che l'istruzione timer è abilitata.
TT	BOOL	Uscita di esecuzione timer. Quando è impostato, è in corso un'operazione di rilevamento del tempo.

Parametro di ingresso	Tipo di dati	Descrizione
DN	BOOL	Uscita di fine rilevazione tempo. Indica quando il tempo accumulato è maggiore di o uguale al valore preimpostato.
Status	DINT	Stato del blocco funzione.
InstructFault (Status.0)	BOOL	L'istruzione ha rilevato uno dei seguenti errori di esecuzione. Non si tratta di un errore minore o grave del controllore. Verificare gli altri bit di stato per stabilire cosa è accaduto.
PresetInv (Status.1)	BOOL	Il valore preimpostato non è valido.

Descrizione: L'istruzione TOFR calcola il tempo finché:

- l'istruzione TOFR non viene disabilitata
- ACC ≥ PRE

La base tempo è sempre 1 msec. Ad esempio, per un timer da 2 secondi, immettere 2000 per il valore .PRE.



Impostare il parametro di ingresso Reset per reimpostare l'istruzione. Se TimerEnable è azzerato quando Reset viene impostato, l'istruzione TOFR non riavvia il rilevamento del tempo quando Reset viene azzerato.

Un timer funziona sottraendo il tempo dell'ultima scansione dal tempo corrente:

ACC = ACC + (*tempo_corrente* – *tempo_ultima_scansione*)

Una volta aggiornato il valore ACC, il timer imposta *tempo_ultima_scansione* = *tempo_corrente*. In questo modo il timer è pronto per la scansione successiva.

IMPORTANTE

Accertarsi di scansionare il timer almeno ogni 69 minuti quando è in funzione. Diversamente, il valore ACC non sarà corretto.

Il valore *tempo_ultima_scansione* ha un intervallo massimo di 69 minuti. Il calcolo del timer ricomincia daccapo se non viene effettuata la scansione entro i 69 minuti. In questo caso, il valore ACC non sarà corretto.

Mentre un timer è in esecuzione, eseguire la scansione entro 69 minuti se lo si colloca all'interno di:

- subroutine
- una sezione di codice compresa tra le istruzioni JMP e LBL
- un diagramma funzionale sequenziale (SFC)
- un evento o task periodico
- una routine di stato di una fase

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

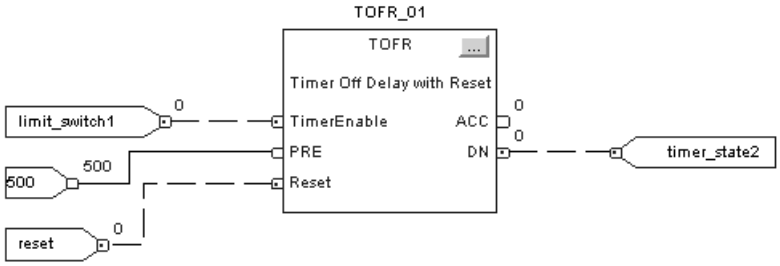
Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Nessuna.	Nessuna.
prima scansione dell'istruzione	EN, TT e DN vengono azzerati. Il valore ACC viene impostato a PRE.	EN, TT e DN vengono azzerati. Il valore ACC viene impostato a PRE.
durante la prima esecuzione dell'istruzione	EN, TT e DN vengono azzerati. Il valore ACC viene impostato a PRE.	EN, TT e DN vengono azzerati. Il valore ACC viene impostato a PRE.
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.	na
EnableIn viene impostato	Quando EnableIn passa da azzerato a impostato, l'istruzione viene inizializzata come descritto per la prima scansione dell'istruzione. L'istruzione viene eseguita. EnableOut viene impostato.	EnableIn viene sempre impostato. L'istruzione viene eseguita.
reset	Quando il parametro di ingresso Reset viene impostato, l'istruzione azzera EN, TT e DN e imposta ACC = PRE. Si noti che questo è diverso dall'uso di un'istruzione RES su un'istruzione TOF.	Quando il parametro di ingresso Reset viene impostato, l'istruzione azzera EN, TT e DN e imposta ACC = PRE. Si noti che questo è diverso dall'uso di un'istruzione RES su un'istruzione TOF.
post-scansione	Nessuna.	Nessuna.

Esempio: A ogni scansione dopo che *limit_switch1* è stato azzerato, l'istruzione TOFR incrementa il valore ACC con il tempo trascorso finché il valore ACC non raggiunge il valore PRE. Quando ACC ≥ PRE, il parametro DN viene azzerato e *timer_state2* viene impostato.

Testo strutturato

```
TOFR_01.Preset := 500
TOFR_01.Reset := reset;
TOFR_O1.TimerEnable := limit_switch1;

TOFR(TOFR_01);

timer_state2 := TOFR_01.DN;
```

Blocco funzione



Retentive Timer On with Reset (RTOR)

L'istruzione RTOR è un timer ritentivo che calcola il tempo quando TimerEnable è impostato.

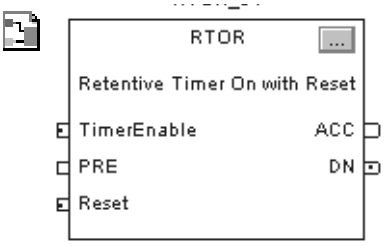
Questa istruzione è disponibile in ladder sotto forma di due istruzioni separate: RTO (vedere pagina 110) e RES (vedere pagina 139).

Operandi:

```
RTOR(RTOR_tag);
```

Testo strutturato

Variabile	Tipo	Formato	Descrizione
tag RTOR	FBD_TIMER	struttura	struttura RTOR



Operandi blocco funzione

Operando	Tipo	Formato	Descrizione
tag RTOR	FBD_TIMER	struttura	struttura RTOR

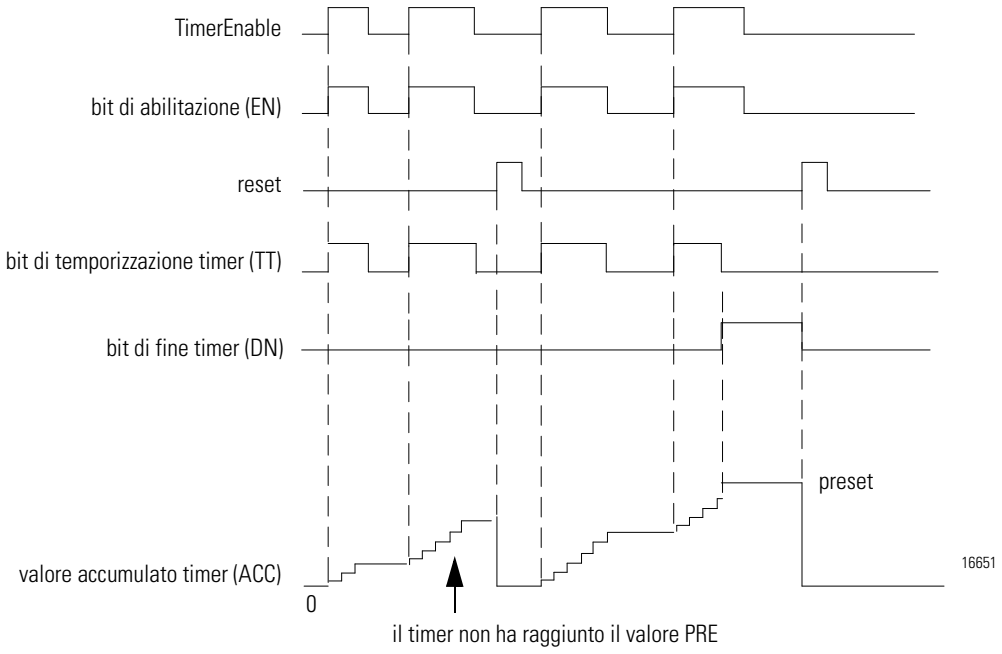
Struttura FBD_TIMER

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato:** nessun effetto. L'istruzione viene eseguita.
TimerEnable	BOOL	Se è impostato, il timer viene eseguito e conteggia il tempo. Il valore di default è azzerato.
PRE	DINT	Valore preimpostato del timer. Si tratta del valore, in unità di 1 msec, che ACC deve raggiungere prima che il rilevamento del tempo venga terminato. Se non è valido, l'istruzione imposta il bit adeguato in Status e il timer non viene eseguito. Valido = da 0 al massimo intero positivo
reset	BOOL	Richiesta di reimpostazione del timer. Quando è impostato, il timer viene reimpostato.
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
ACC	DINT	Tempo accumulato in millisecondi. Questo valore viene mantenuto anche mentre l'ingresso TimerEnable è azzerato. Ciò rende il comportamento di questo blocco diverso da quello del blocco TONR.
EN	BOOL	Uscita di abilitazione timer. Indica che l'istruzione timer è abilitata.
TT	BOOL	Uscita di esecuzione timer. Quando è impostato, è in corso un'operazione di rilevamento del tempo.

Parametro di ingresso	Tipo di dati	Descrizione
DN	BOOL	Uscita di fine rilevazione tempo. Indica quando il tempo accumulato è maggiore di o uguale al valore preimpostato.
Status	DINT	Stato del blocco funzione.
InstructFault (Status.0)	BOOL	L'istruzione ha rilevato uno dei seguenti errori di esecuzione. Non si tratta di un errore minore o grave del controllore. Verificare gli altri bit di stato per stabilire cosa è accaduto.
PresetInv (Status.1)	BOOL	Il valore preimpostato non è valido.

Descrizione: L'istruzione RTOR calcola il tempo finché non viene disabilitata. Quando è disabilitata, l'istruzione RTOR conserva il proprio valore ACC. È necessario azzerare il valore .ACC utilizzando l'ingresso Reset.

La base tempo è sempre 1 msec. Ad esempio, per un timer da 2 secondi, immettere 2000 per il valore .PRE.



Impostare il parametro di ingresso Reset per reimpostare l'istruzione. Se TimerEnable è impostato quando Reset viene impostato, l'istruzione RTOR inizia a rilevare di nuovo il tempo quando Reset viene azzerato.

Un timer funziona sottraendo il tempo dell'ultima scansione dal tempo corrente:

ACC = ACC + (*tempo_corrente* – *tempo_ultima_scansione*)

Una volta aggiornato il valore ACC, il timer imposta *tempo_ultima_scansione* = *tempo_corrente*. In questo modo il timer è pronto per la scansione successiva.

IMPORTANTE

Accertarsi di scansionare il timer almeno ogni 69 minuti quando è in funzione. Diversamente, il valore ACC non sarà corretto.

Il valore *tempo_ultima_scansione* ha un intervallo massimo di 69 minuti. Il calcolo del timer ricomincia daccapo se non viene effettuata la scansione entro i 69 minuti. In questo caso, il valore ACC non sarà corretto.

Mentre un timer è in esecuzione, eseguire la scansione entro 69 minuti se lo si colloca all'interno di:

- subroutine
- una sezione di codice compresa tra le istruzioni JMP e LBL
- un diagramma funzionale sequenziale (SFC)
- un evento o task periodico
- una routine di stato di una fase

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

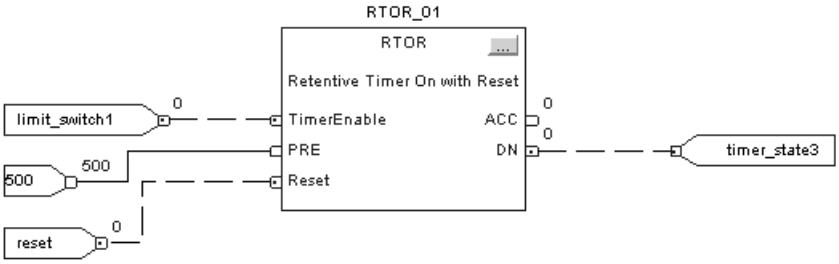
Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Nessuna.	Nessuna.
prima scansione dell'istruzione	EN, TT e DN vengono azzerati Il valore ACC non viene modificato	EN, TT e DN vengono azzerati Il valore ACC non viene modificato
durante la prima esecuzione dell'istruzione	EN, TT e DN vengono azzerati Il valore ACC non viene modificato	EN, TT e DN vengono azzerati Il valore ACC non viene modificato
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.	na
EnableIn viene impostato	**Blocco funzione:** Quando EnableIn passa da azzerato a impostato, l'istruzione viene inizializzata come descritto per la prima scansione dell'istruzione. L'istruzione viene eseguita. EnableOut viene impostato.	EnableIn viene sempre impostato. L'istruzione viene eseguita.
reset	Quando il parametro di ingresso Reset viene impostato, l'istruzione azzera EN, TT e DN e imposta ACC = zero.	Quando il parametro di ingresso Reset viene impostato, l'istruzione azzera EN, TT e DN e imposta ACC = zero.
post-scansione	Nessuna.	Nessuna.

Esempio: A ogni scansione in cui *limit_switch1* è impostato, l'istruzione RTOR incrementa il valore ACC con il tempo trascorso finché il valore ACC non raggiunge il valore PRE. Quando ACC ≥ PRE, il parametro DN e *timer_state3* vengono impostati.

Testo strutturato

```
RTOR_01.Preset := 500
RTOR_01.Reset := reset;
RTOR_O1.TimerEnable := limit_switch1;

RTOR(RTOR_01);

timer_state3 := RTOR_01.DN;
```

Blocco funzione



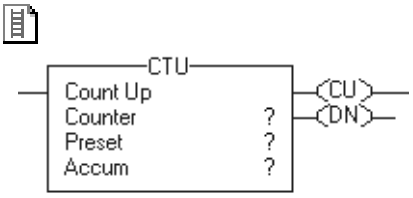
Count Up (CTU)

L'istruzione CTU conteggia in modo incrementale.

L'istruzione è disponibile in testo strutturato e blocco funzione come CTUD, vedere pagina 134.

Operandi:

Ladder

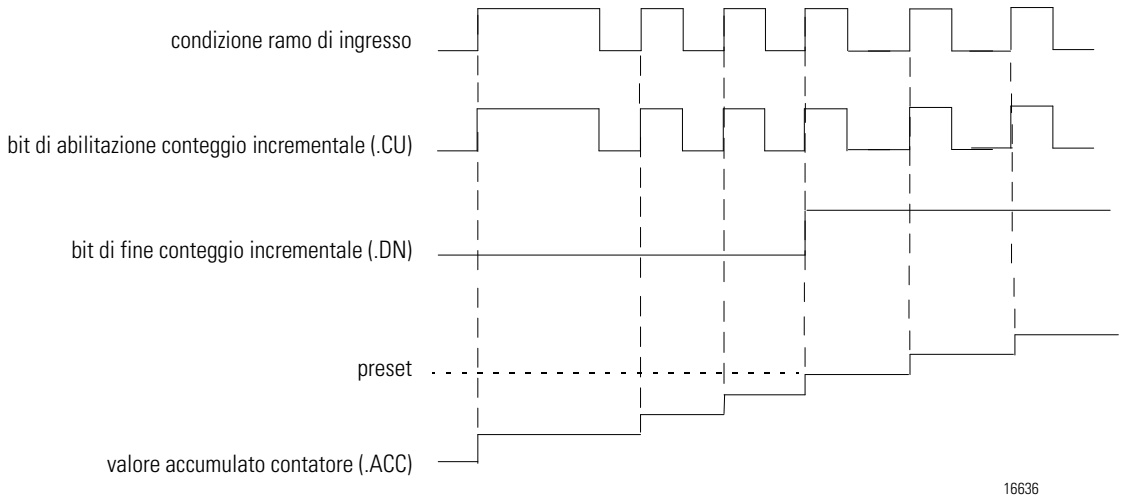


Operando	Tipo	Formato	Descrizione
Counter	COUNTER	tag	struttura COUNTER
Preset	DINT	immediato	limite massimo del conteggio
Accum	DINT	immediato	numero di volte calcolate dal contatore il valore iniziale generalmente è 0

Struttura COUNTER

Mnemonico	Tipo di dati	Descrizione
.CU	BOOL	Il bit di abilitazione del conteggio incrementale indica che l'istruzione CTU è abilitata.
.DN	BOOL	Il bit di fine indica che .ACC ≥ .PRE.
.OV	BOOL	Il bit di overflow indica che il contatore ha superato il limite massimo di 2 147 483 647. Il contatore continua quindi a contare in modo incrementale a partire da –2 147 483 648.
.UN	BOOL	Il bit di underflow indica che il contatore ha superato il limite inferiore di –2 147 483 648. Il contatore continua quindi a contare in modo decrementale a partire da 2 147 483 647.
.PRE	DINT	Il valore preimpostato specifica il valore che il valore accumulato deve raggiungere prima che l'istruzione imposti il bit .DN.
.ACC	DINT	Il valore accumulato indica il numero di transizioni contate dall'istruzione.

Descrizione: Quando è abilitata e il bit .CU è azzerato, l'istruzione CTU incrementa il contatore di uno. Quando è abilitata e il bit .CU è impostato, o quanto è disabilitata, l'istruzione CTU conserva il proprio valore .ACC.

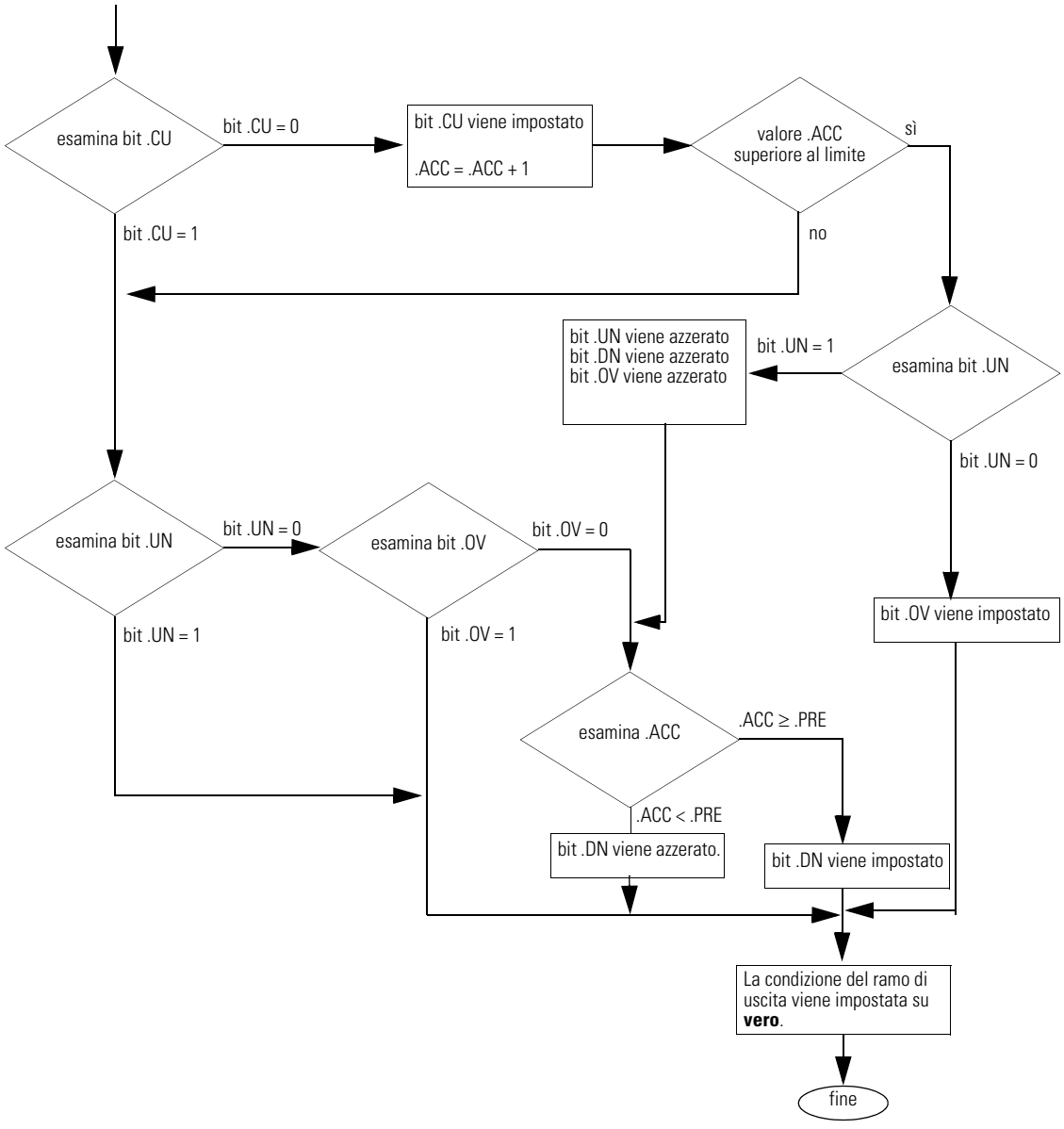


Il valore accumulato continua ad aumentare, anche dopo l'impostazione del bit .DN. Per azzerare il valore accumulato, utilizzare un'istruzione RES che faccia riferimento alla struttura del contatore o scriva 0 sul valore accumulato.

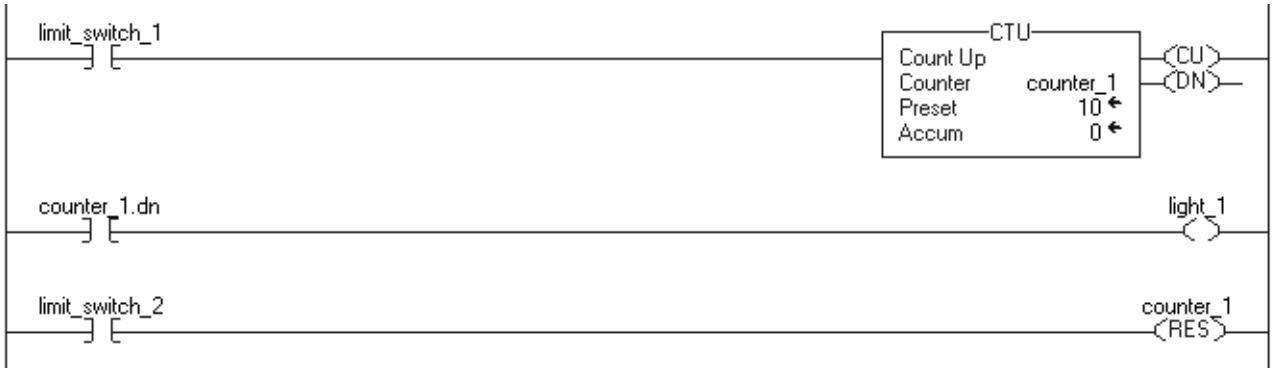
Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit .CU viene impostato per evitare incrementi non validi durante la prima scansione del programma. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit .CU viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Dopo che *limit_switch_1* è passato da disabilitato ad abilitato 10 volte, il bit .DN viene impostato e *light_1* si accende. Se *limit_switch_1* continua a passare da disabilitato ad abilitato, *counter_1* continua ad incrementare il proprio conteggio e il bit .DN rimane impostato. Quando limit_switch_2 è abilitato, l'istruzione RES reimposta *counter_1* (azzera i bit di stato e il valore .ACC) e *light_1* si spegne.



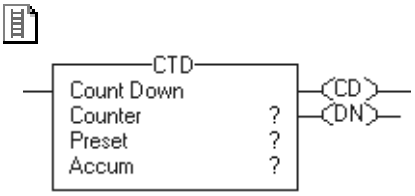
Count Down (CTD)

L'istruzione CTD conta in modo decrementale.

L'istruzione è disponibile in testo strutturato e blocco funzione come CTUD, vedere pagina 134.

Operandi:

Ladder



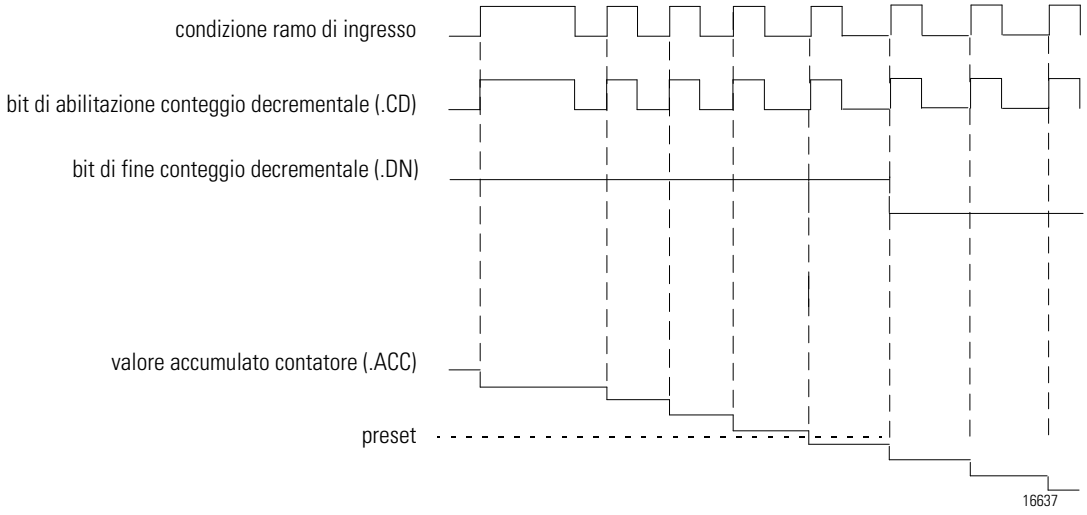
Operando	Tipo	Formato	Descrizione
Counter	COUNTER	tag	struttura COUNTER
Preset	DINT	immediato	limite minimo del conteggio
Accum	DINT	immediato	numero di volte calcolate dal contatore il valore iniziale generalmente è 0

Struttura COUNTER

Mnemonico	Tipo di dati	Descrizione
.CD	BOOL	Il bit di abilitazione del conteggio decrementale indica che l'istruzione CTD è abilitata.
.DN	BOOL	Il bit di fine indica che .ACC ≥ .PRE.
.OV	BOOL	Il bit di overflow indica che il contatore ha superato il limite massimo di 2 147 483 647. Il contatore continua quindi a contare in modo incrementale a partire da –2 147 483 648.
.UN	BOOL	Il bit di underflow indica che il contatore ha superato il limite inferiore di –2 147 483 648. Il contatore continua quindi a contare in modo decrementale a partire da 2 147 483 647.
.PRE	DINT	Il valore preimpostato specifica il valore che il valore accumulato deve raggiungere prima che l'istruzione imposti il bit .DN.
.ACC	DINT	Il valore accumulato indica il numero di transizioni contate dall'istruzione.

Descrizione: L'istruzione CTD viene di norma usata con un'istruzione CTU che fa riferimento alla stessa struttura COUNTER.

Quando è abilitata e il bit .CD è azzerato, l'istruzione CTD decrementa il contatore di uno. Quando è abilitata e il bit .CD è impostato, o quanto è disabilitata, l'istruzione CTD conserva il proprio valore .ACC.

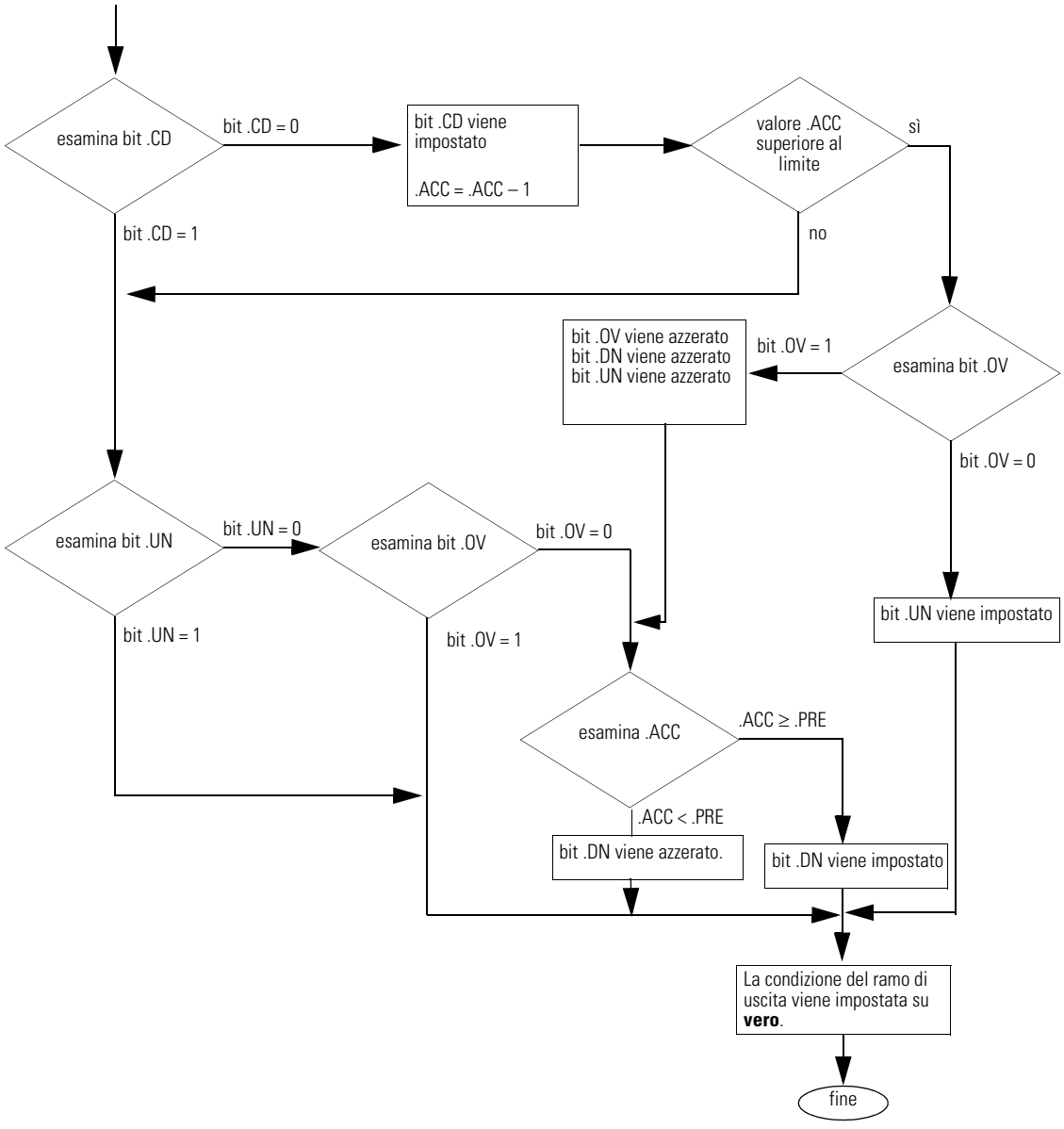


Il valore accumulato continua a diminuire, anche dopo l'impostazione del bit .DN. Per azzerare il valore accumulato, utilizzare un'istruzione RES che faccia riferimento alla struttura del contatore o scriva 0 sul valore accumulato.

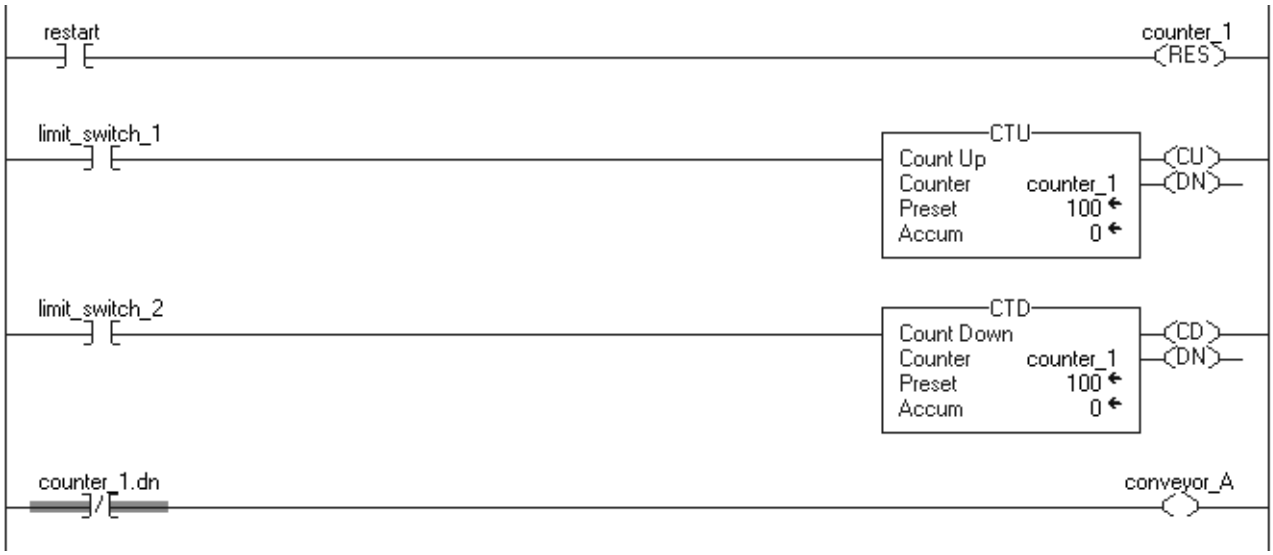
Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit .CD viene impostato per evitare decrementi non validi durante la prima scansione del programma. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit .CD viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Un nastro trasportatore porta componenti in una zona di stoccaggio. Ogni volta che viene immesso un componente, *limit_switch_1* viene abilitato e *counter_1* viene incrementato di 1. Ogni volta che un componente esce dal nastro trasportatore, *limit_switch_2* viene abilitato e *counter_1* viene decrementato di 1. Se nella zona di stoccaggio vi sono 100 componenti (*counter_1.dn* è impostato), *conveyor_a* viene attivato e impedisce al nastro trasportatore di portare altri componenti finché la zona di stoccaggio non ha altro spazio.



Count Up/Down (CTUD)

L'istruzione CTUD conta a incrementi di 1 quando CUEnable passa da azzerato a impostato. L'istruzione conta a decrementi di 1 quando CDEnable passa da azzerato a impostato.

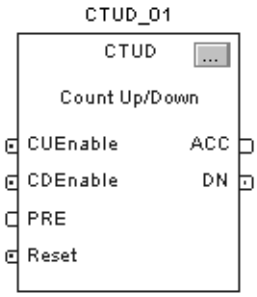
Questa istruzione è disponibile in ladder sotto forma di tre istruzioni separate: CTU (vedere pagina 126), CTD (vedere pagina 130) e RES (vedere pagina 139).

Operandi:

```
CTUD(CTUD_tag);
```

Testo strutturato

Variabile	Tipo	Formato	Descrizione
tag CTUD	FBD_COUNTER	struttura	struttura CTUD



Blocco funzione

Operando	Tipo	Formato	Descrizione
tag CTUD	FBD_COUNTER	struttura	struttura CTUD

Struttura FBD_COUNTER

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato:** Nessun effetto. L'istruzione viene eseguita.
CUEnable	BOOL	Abilita il conteggio incrementale. Quando l'ingresso passa da azzerato a impostato, il contatore conta a incrementi di 1. Il valore di default è azzerato.

Parametro di ingresso	Tipo di dati	Descrizione
CDEnable	BOOL	Abilita il conteggio decrementale. Quando l'ingresso passa da azzerato a impostato, il contatore conta a decrementi di 1. Il valore di default è azzerato.
PRE	DINT	Valore preimpostato del contatore. Si tratta del valore che il valore accumulato deve raggiungere prima che venga impostato DN. Valori validi = qualsiasi numero intero Il valore di default è 0.
Reset	BOOL	Richiesta di reimpostazione del timer. Quando impostato, il contatore viene reimpostato. Il valore di default è azzerato.

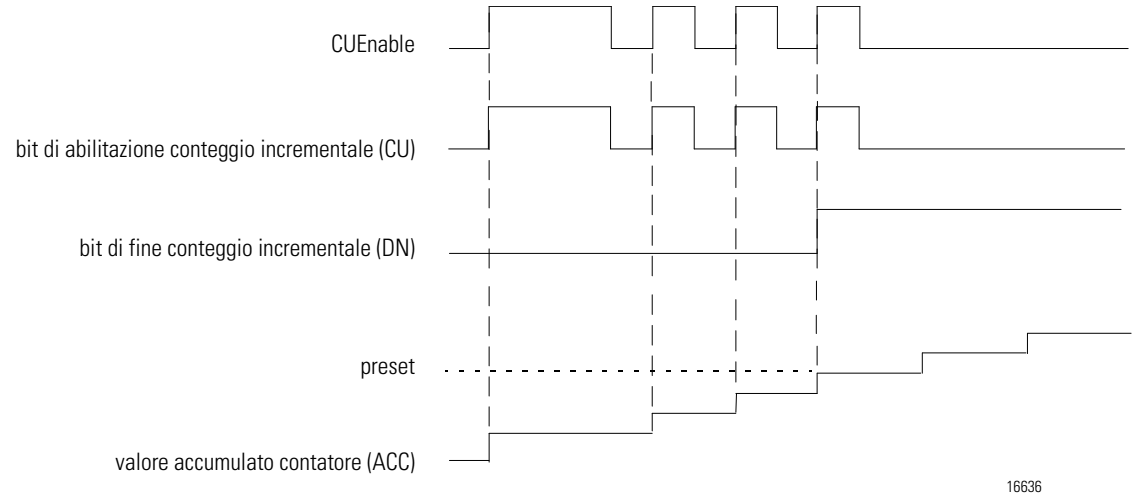
Parametro di uscita	Tipo di dati	Descrizione
EnableOut	BOOL	L'istruzione produce un risultato valido.
ACC	DINT	Valore accumulato.
CU	BOOL	Conteggio incrementale abilitato.
CD	BOOL	Conteggio decrementale abilitato.
DN	BOOL	Conteggio terminato. Impostato quando il valore accumulato è maggiore di o uguale al valore preimpostato.
OV	BOOL	Overflow contatore. Indica che il contatore ha superato il limite massimo di 2 147 483 647. Il contatore continua quindi a contare in modo decrementale a partire da –2 147 483 648.
UN	BOOL	Underflow contatore. Indica che il contatore ha superato il limite minimo di –2 147 483 648. Il contatore continua quindi a contare in modo decrementale a partire da 2 147 483 647.

Descrizione

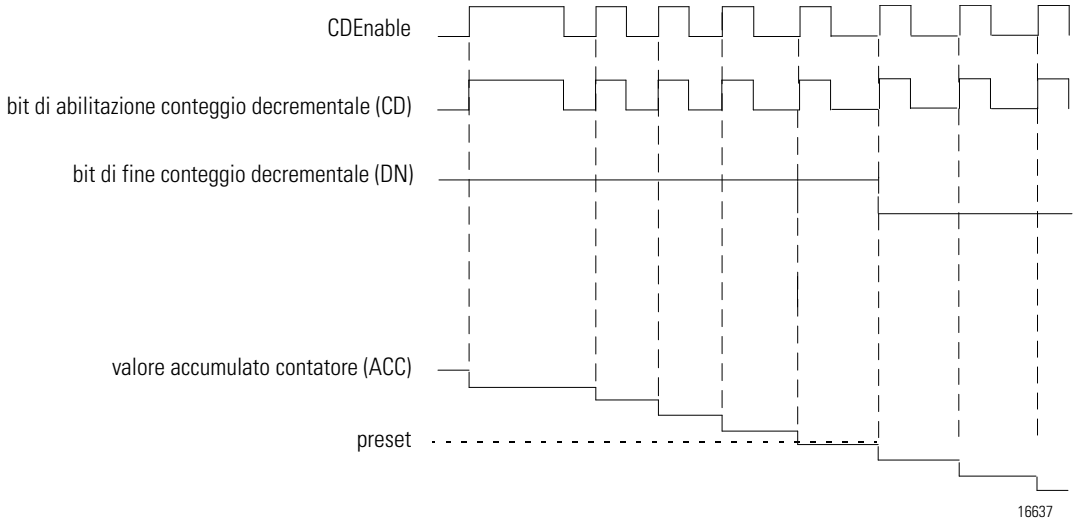
Quando è abilitata e CUEnable è impostato, l'istruzione CTUD incrementa il contatore di uno. Quando è abilitata e CDEnable è impostato, l'istruzione CTUD decrementa il contatore di uno.

I parametri di ingresso CUEnable e CDEnable possono entrambi passare da uno stato all'altro durante la stessa scansione. L'istruzione esegue il conteggio incrementale prima del conteggio decrementale.

Conteggio incrementale



Conteggio decrementale



Quando è disabilitata, l'istruzione CTUD conserva il proprio valore accumulato. Impostare il parametro di ingresso Reset della struttura FBD_COUNTER per reimpostare l'istruzione.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

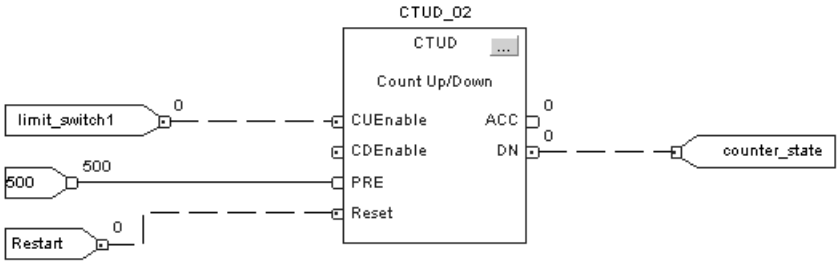
Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Nessuna inizializzazione necessaria.	Nessuna inizializzazione necessaria.
prima scansione dell'istruzione	$CUEnable_{n-1}$ e $CDEnable_{n-1}$ vengono impostati.	$CUEnable_{n-1}$ e $CDEnable_{n-1}$ vengono impostati.
durante la prima esecuzione dell'istruzione	$CUEnable_{n-1}$ e $CDEnable_{n-1}$ vengono impostati.	$CUEnable_{n-1}$ e $CDEnable_{n-1}$ vengono impostati.
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.	na
EnableIn viene impostato	L'istruzione imposta $CUEnable_{n-1}$ e $CDEnable_{n-1}$. In una transizione azzerato-impostato di EnableIn: • L'istruzione viene eseguita. • EnableOut viene impostato.	L'istruzione imposta $CUEnable_{n-1}$ e $CDEnable_{n-1}$. EnableIn viene sempre impostato. L'istruzione viene eseguita.
reset	Quando impostato, l'istruzione azzera $CUEnable_{n-1}$, $CDEnable_{n-1}$, CU, CD, DN, OV e UN e imposta ACC = zero.	Quando impostato, l'istruzione azzera $CUEnable_{n-1}$, $CDEnable_{n-1}$, CU, CD, DN, OV e UN e imposta ACC = zero.
post-scansione	Nessuna.	Nessuna.

Esempio: Quando *limit_switch1* passa da azzerato a impostato, CUEnable viene impostato per una scansione e l'istruzione CTUD incrementa il valore ACC di 1. Quando ACC ≥ PRE, il parametro DN viene impostato, abilitando l'istruzione di blocco funzione che segue l'istruzione CTUD.

Testo strutturato

```
CTUD_01.Preset := 500;
CTUD_01.Reset := Restart;
CTUD_O1.CUEnable := limit_switch1;

CTUD(CTUD_01);

counter_state := CTUD_01.DN;
```

Blocco funzione

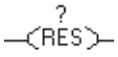

CTUD_02
CTUD
...
Count Up/Down
limit_switch1
0
CUEnable
ACC
0
CDEnable
DN
0
counter_state
500
500
PRE
Reset
Restart
0

Reset (RES)

L'istruzione RES reimposta una struttura TIMER, COUNTER o CONTROL

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
struttura	TIMER CONTROL COUNTER	tag	struttura da reimpostare

Descrizione:

Quando è abilitata, l'istruzione RES azzera i seguenti elementi:

Quando si utilizza un'istruzione RES per	L'istruzione azzera
TIMER	valore .ACC i bit di stato di controllo
COUNTER	valore .ACC i bit di stato di controllo
CONTROL	valore .POS i bit di stato di controllo

ATTENZIONE

Dato che l'istruzione RES azzera il valore .ACC, il bit .DN e il bit .TT, non utilizzare l'istruzione RES per reimpostare un timer TOF.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione RES azzera la struttura specificata. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Istruzioni di I/O
(MSG, GSV, SSV, IOT)

Introduzione

Le istruzioni di input/output leggono o scrivono dati sul/dal controllore o blocchi di dati a/da un altro modulo su un'altra rete.

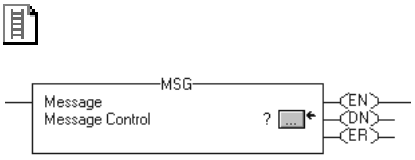
Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
inviare/ricevere dati a/da un altro modulo	MSG	ladder testo strutturato	142
ottenere informazioni sullo stato del controllore	GSV	ladder testo strutturato	173
impostare informazioni sullo stato del controllore	SSV	ladder testo strutturato	173
• inviare valori di uscita a un modulo I/O o al controllore consumatore in un punto specifico della logica • attivare un task evento in un altro controllore	IOT	ladder testo strutturato	198

Message (MSG)

L'istruzione MSG legge o scrive un blocco di dati ad un altro modulo di una rete in modo asincrono.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Message control	MESSAGE	tag	struttura messaggio

Testo strutturato

```
MSG(MessageControl);
```

Gli operandi sono uguali a quelli utilizzati per l'istruzione MSG ladder.

Struttura MESSAGE

ATTENZIONE

Se i bit di stato vengono controllati più di una volta

Il controllore modifica i bit DN, ER, EW e ST in maniera asincrona nella scansione della logica. Usare una copia dei bit se questi vengono controllati in più punti all'interno della logica. Diversamente, i bit potrebbero essere modificati durante la scansione e la logica non funzionerà nel modo previsto.

Un modo per creare una copia dei bit è utilizzare la parola FLAGS. Copiare la parola FLAGS in un altro tag e verificare i bit della copia.

IMPORTANTE

Non modificare i seguenti bit di stato di un'istruzione MSG:

- DN
- EN
- ER
- EW
- ST

Non modificare tali bit né da soli né all'interno di una parola FLAGS. In tal caso, il controllore potrebbe subire un errore irreversibile. In caso di errore irreversibile, il controllore elimina il progetto dalla memoria.

Mnemonico	Tipo di dati	Descrizione
.FLAGS	INT	Il membro .FLAGS permette di accedere ai membri di stato (bit) in una parola a 16 bit. **Il bit: È il membro:** 2 .EW 4 .ER 5 .DN 6 .ST 7 .EN 8 .TO 9 .EN_CC **Importante:** non modificare i bit EW, ER, DN o ST del membro FLAGS. Ad esempio, non azzerare l'intera parola FLAGS. Il controllore ignora la modifica e utilizza i valori dei bit memorizzati internamente.
.ERR	INT	Se il bit .ER è impostato, la parola codice di errore segnala i codici di errore dell'istruzione MSG.
.EXERR	INT	La parola codice di errore esteso segnala informazioni aggiuntive per alcuni codici di errore.
.REQ_LEN	INT	La lunghezza richiesta indica quante parole l'istruzione di messaggio tenterà di trasferire.
.DN_LEN	INT	La lunghezza completata indica quante parole sono state effettivamente trasferite.
.EW	BOOL	Il bit di abilitazione attesa viene impostato quando il controllore rileva che è stata messa in coda una richiesta di messaggio. Il controllore azzera il bit .EW quando viene impostato il bit .ST. **Importante:** non modificare il bit EW. Il controllore ignora la modifica e utilizza i valori del bit memorizzati internamente.
.ER	BOOL	Il bit di errore viene impostato quando il controllore rileva un trasferimento errato. Il bit .ER viene azzerato la volta successiva che la condizione del ramo di ingresso passa da falsa a vera. **Importante:** non modificare il bit ER.
.DN	BOOL	Il bit di fine viene impostato quando l'ultimo pacchetto del messaggio viene trasferito senza errori. Il bit .DN viene azzerato la volta successiva che la condizione del ramo di ingresso passa da falsa a vera. **Importante:** non modificare il bit DN.
.ST	BOOL	Il bit di inizio viene impostato quando il controllore inizia ad eseguire l'istruzione MSG. Il bit .ST viene azzerato quando viene impostato il bit .DN o il bit .ER. **Importante:** non modificare il bit ST. Il controllore ignora la modifica e utilizza i valori del bit memorizzati internamente.
.EN	BOOL	Il bit di abilitazione viene impostato quando la condizione del ramo di ingresso diventa vera e rimane impostato fino a quando non viene impostato il bit .DN o il bit .ER e la condizione del ramo di ingresso è falsa. Se la condizione del ramo di ingresso diventa falsa ma il bit .DN ed il bit .ER sono azzerati, il bit .EN rimane impostato. **Importante:** non modificare il bit EN.
.TO	BOOL	Se si imposta manualmente il bit .TO, il controllore smette di elaborare il messaggio ed imposta il bit .ER.
.EN_CC	BOOL	Il bit di abilitazione cache stabilisce come gestire la connessione MSG. Vedere Scegliere un'opzione cache a pagina 4-170. Le connessioni delle istruzioni MSG in uscita dalla porta seriale sono senza cache, anche se il bit .EN_CC è impostato.
.ERR_SRC	SINT	Utilizzato dal software RSLogix 5000 per mostrare il percorso di errore nella finestra di dialogo Message Configuration
.DestinationLink	INT	Per modificare il Collegamento di destinazione di un DH+ o CIP con messaggio di ID origine, impostare questo membro al valore richiesto.

Mnemonico	Tipo di dati	Descrizione
.DestinationNode	INT	Per modificare il Nodo di destinazione di un DH+ o CIP con messaggio di ID origine, impostare questo membro al valore richiesto.
.SourceLink	INT	Per modificare il Collegamento di origine di un DH+ o CIP con messaggio di ID origine, impostare questo membro al valore richiesto.
.Class	INT	Per modificare il parametro Class di un messaggio CIP generico, impostare questo membro al valore richiesto.
.Attribute	INT	Per modificare il parametro Attribute di un messaggio CIP generico, impostare questo membro al valore richiesto.
.Instance	DINT	Per modificare il parametro Instance di un messaggio CIP generico, impostare questo membro al valore richiesto.
.LocalIndex	DINT	Se si usa un asterisco [*] per indicare il numero di elemento della matrice locale, LocalIndex fornisce il numero di elemento. Per modificare il numero di elemento, impostare questo membro al valore richiesto. **Se il messaggio:** / **La matrice locale è:** legge dati / elemento di destinazione scrive dati / elemento di origine
.Channel	SINT	Per inviare il messaggio da un canale diverso del modulo 1756-DHRIO, impostare questo membro al valore richiesto. Utilizzare il carattere ASCII A o B.
.Rack	SINT	Per modificare il numero di rack di un messaggio di trasferimento a blocchi, impostare questo membro al numero di rack richiesto (ottale).
.Group	SINT	Per modificare il numero di gruppo di un messaggio di trasferimento a blocchi, impostare questo membro al numero di gruppo richiesto (ottale).
.Slot	SINT	Per modificare il numero di slot di un messaggio di trasferimento a blocchi, impostare questo membro al numero di slot richiesto. **Se il messaggio è su una rete:** / **Specificare il numero di slot in:** universal remote I/O / Ottale ControlNet / Decimale (0–15)
.Path	STRING	Per inviare il messaggio a un controllore diverso, impostare questo membro al nuovo percorso. • Immettere il percorso in valori esadecimali. • Omettere le virgole [,] Ad esempio, per un percorso di 1, 0, 2, 42, 1, 3, immettere $01$00$02$2A$01$03. Per passare a un dispositivo e creare automaticamente la nuova stringa o parte di essa, fare clic con il pulsante destro del mouse su un tag stringa, quindi scegliere *Go to Message Path Editor*.
.RemoteIndex	DINT	Se si usa un asterisco [*] per indicare il numero di elemento della matrice remota, RemoteIndex fornisce il numero di elemento. Per modificare il numero di elemento, impostare questo membro al valore richiesto. **Se il messaggio:** / **La matrice remota è:** legge dati / elemento di origine scrive dati / elemento di destinazione

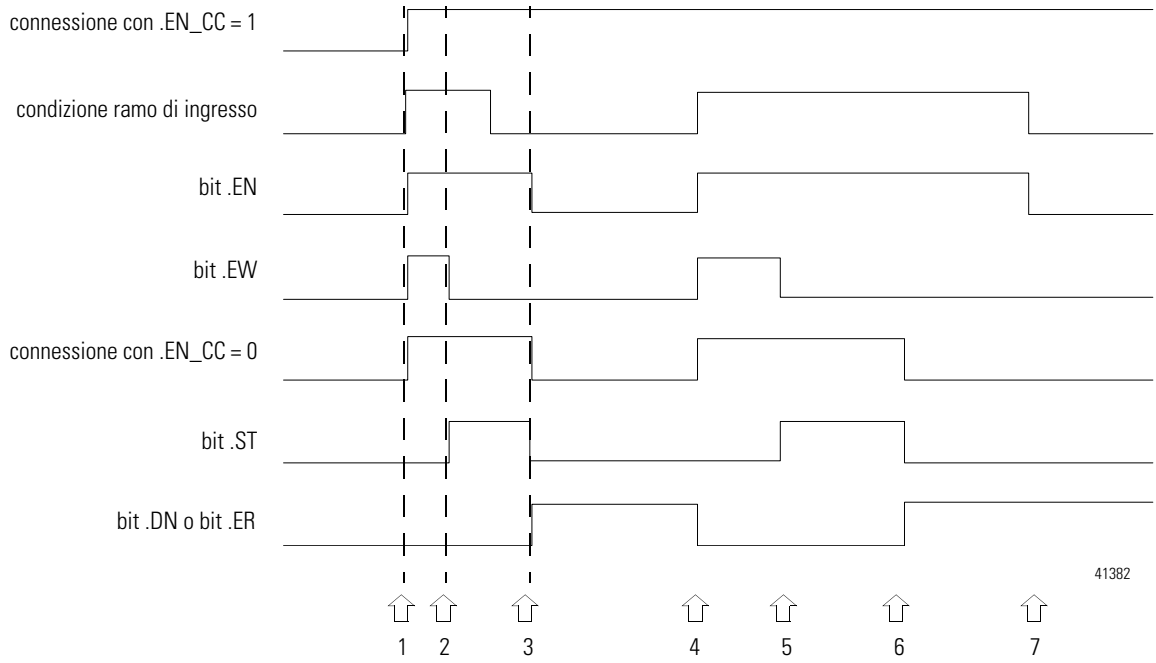
Mnemonico	Tipo di dati	Descrizione	
.RemoteElement	STRING	Per specificare un tag o un indirizzo diverso nel controllore al quale il messaggio viene inviato, impostare questo membro al valore richiesto. Immettere il tag o l'indirizzo in caratteri ASCII.	
		Se il messaggio:	**La matrice remota è:**
		legge dati	elemento di origine
		scrive dati	elemento di destinazione
.UnconnnectedTimeout	DINT	Timeout per assenza di connessione per i messaggi. Il valore predefinito è 30 secondi.	
		Se il messaggio è:	**Allora:**
		non connesso	Il bit ER si attiva se il controllore non riceve una risposta entro il tempo UnconnectedTimeout.
		connesso	Il bit ER si attiva se il controllore non riceve una risposta per la realizzazione della connessione entro il tempo UnconnectedTimeout.
.ConnectionRate	DINT	Timeout per un messaggio connesso una volta che ha una connessione. Questo timeout è per la risposta proveniente dall'altro dispositivo in merito all'invio dei dati. • Questo timeout si applica solo una volta eseguita la connessione. • Timeout = ConnectionRate x TimeoutMultiplier. • Il valore predefinito di ConnectionRate è 7,5 secondi. • Il valore predefinito di TimeoutMultiplier è 0 (equivalente a un fattore di moltiplicazione di 4). • Il time out predefinito per la connessione per i messaggi è 30 secondi (7,5 secondi x 4 = 30 secondi). • Per modificare il time out, modificare ConnectionRate e lasciare TimeoutMultiplier al valore predefinito.	
.TimeoutMultiplier	SINT		

Descrizione L'istruzione MSG trasferisce elementi di dati.

Si tratta di un'istruzione di transizione:

- In ladder, cambiare la condizione del ramo di ingresso da azzerata a impostata per ogni esecuzione dell'istruzione richiesta.
- In testo strutturato, condizionare l'istruzione in modo che venga eseguita solo in caso di transizione. Vedere l'Appendice B.

La dimensione di ciascun elemento dipende dal tipo di dati specificato e dal tipo di comando di messaggio utilizzato.

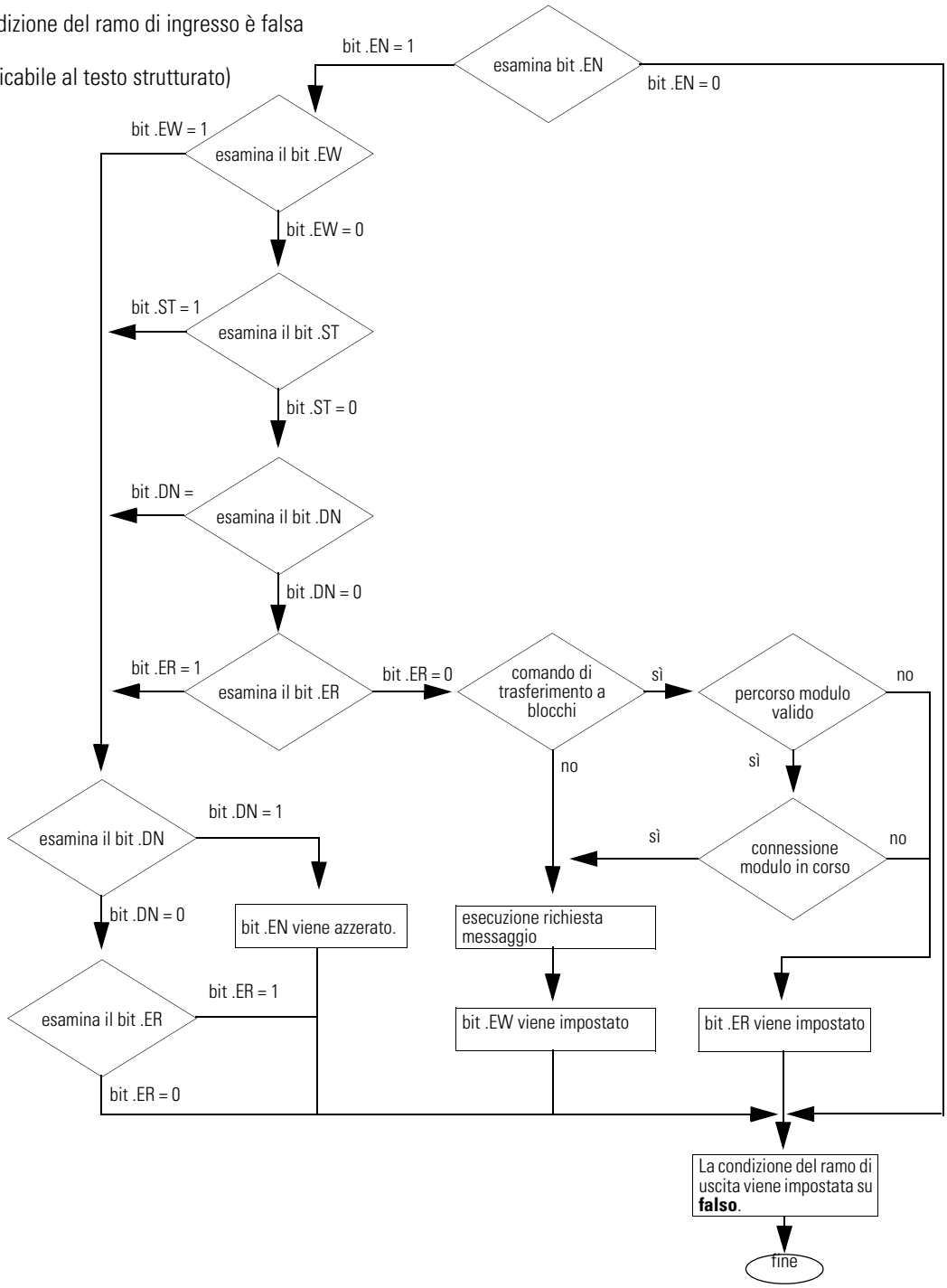


Dove	Descrizione	Dove	Descrizione
1	se la condizione del ramo di ingresso è vera .EN viene impostato .EW viene impostato la connessione è aperta*	5	il messaggio è inviato .ST viene impostato .EW viene azzerato
2	il messaggio è inviato .ST viene impostato .EW viene azzerato	6	messaggio terminato o con errori la condizione del ramo di ingresso è ancora vera .DN o .ER viene impostato .ST è azzerato la connessione viene chiusa (se .EN_CC = 0)
3	messaggio terminato o con errori se la condizione del ramo di ingresso è falsa .DN o .ER viene impostato .ST è azzerato la connessione viene chiusa (se .EN_CC = 0) .EN viene azzerato (condizione del ramo di ingresso falsa)	7	la condizione del ramo di ingresso diventa falsa e .DN o .ER è impostato .EN viene azzerato.

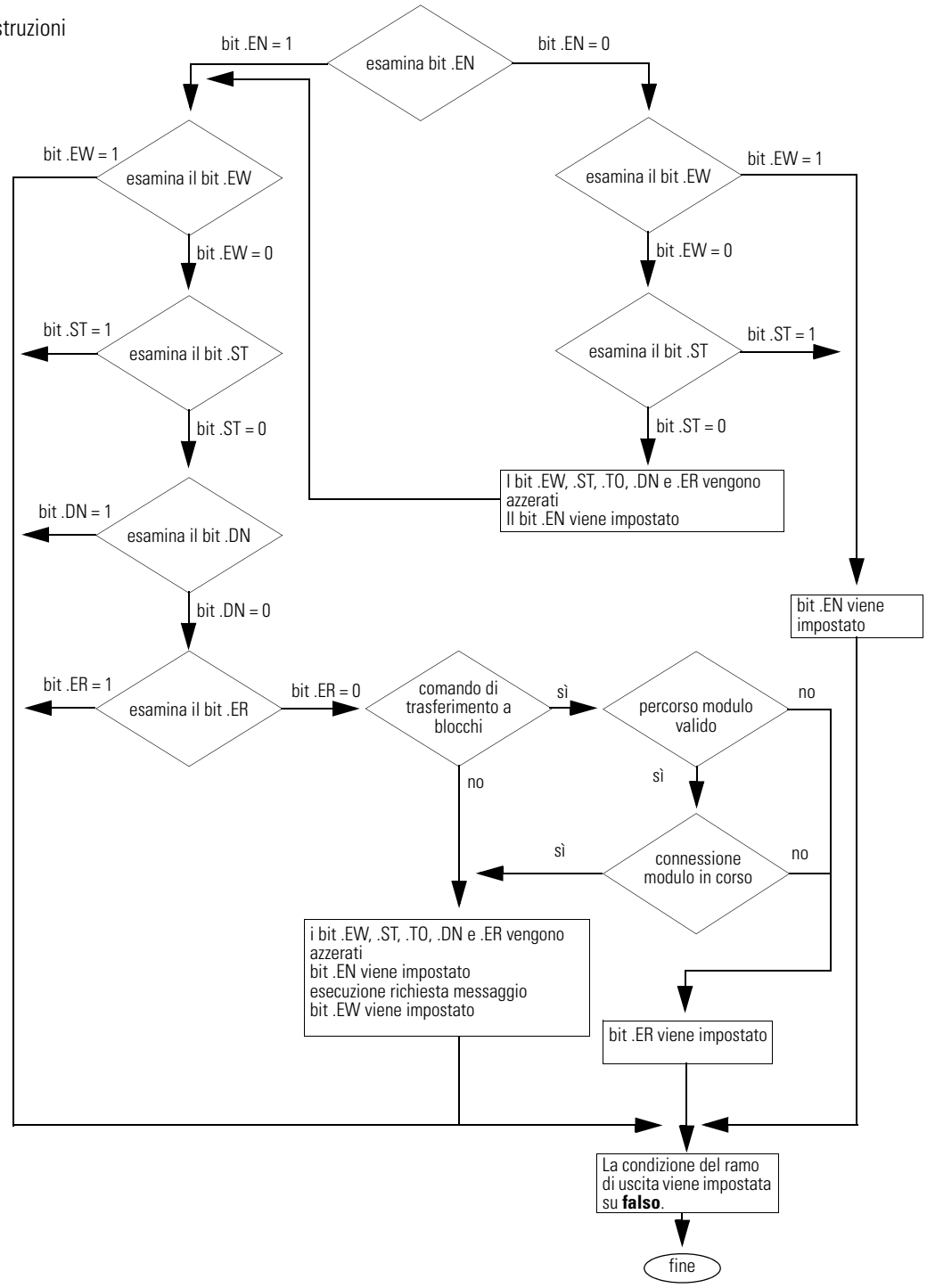
Dove	Descrizione	Dove	Descrizione
4	se la condizione del ramo di ingresso è vera .DN o .ER è stato impostato precedentemente .EN viene impostato .EW viene impostato la connessione è aperta* .DN o .ER viene azzerato		

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Condizione	Azione ladder	Azione testo strutturato
se la condizione del ramo di ingresso è falsa (non applicabile al testo strutturato)		
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na

Condizione	Azione ladder	Azione testo strutturato
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.

Condizione	Azione ladder	Azione testo strutturato
esecuzione istruzioni		
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Codici di errore MSG

I codici di errore dipendono dal tipo di istruzione MSG.

Codici di errore

Il software RSLogix 5000 non sempre visualizza l'intera descrizione.

Codice di errore (esad.)	Descrizione	Visualizzazione nel software
0001	Collegamento interrotto (vedere codici di errore estesi)	come nella descrizione
0002	Risorse insufficienti	come nella descrizione
0003	Valore non valido	come nella descrizione
0004	Errore di sintassi IOI (vedere codici di errore estesi)	come nella descrizione
0005	Destinazione sconosciuta, classe non supportata, istanza non definita o elemento struttura non definito (vedere codici di errore estesi)	come nella descrizione
0006	Spazio pacchetto insufficiente	come nella descrizione
0007	Connessione interrotta	come nella descrizione
0008	Servizio non supportato	come nella descrizione
0009	Errore nel segmento dati o valore attributo non valido	come nella descrizione
000A	Errore elenco attributi	come nella descrizione
000B	Stato già esistente	come nella descrizione
000C	Conflitto modello oggetto	come nella descrizione
000D	Oggetto già esistente	come nella descrizione
000E	Impossibile impostare l'attributo	come nella descrizione
000F	Permesso negato	come nella descrizione
0010	Conflitto di stato dispositivo	come nella descrizione
0011	Risposta non adatta	come nella descrizione
0012	Frammento primario	come nella descrizione
0013	Dati di comando insufficienti	come nella descrizione
0014	Attributo non supportato	come nella descrizione
0015	Troppi dati	come nella descrizione
001A	Richiesta ponte troppo grande	come nella descrizione
001B	Risposta ponte troppo grande	come nella descrizione
001C	Elenco attributi vuoto	come nella descrizione
001D	Elenco attributi non valido	come nella descrizione
001E	Errore di servizio integrato	come nella descrizione
001F	Errore di connessione correlata (vedere codici di errore estesi)	come nella descrizione
0022	È stata ricevuta una risposta non valida	come nella descrizione
0025	Errore segmento chiave	come nella descrizione
0026	Errore di IOI non valido	come nella descrizione
0027	Attributo non previsto in elenco	come nella descrizione
0028	Errore DeviceNet – ID membro non valida	come nella descrizione

Codice di errore (esad.)	Descrizione	Visualizzazione nel software
0029	Errore DeviceNet – impossibile impostare il membro	come nella descrizione
00D1	Il modulo non è in stato di esecuzione	Errore sconosciuto
00FB	Porta messaggi non supportata	Errore sconosciuto
00FC	Il messaggio non supporta il tipo di dati	Errore sconosciuto
00FD	Messaggio non inizializzato	Errore sconosciuto
00FE	Timeout del messaggio	Errore sconosciuto
00FF	Errore generale (vedere codici di errore estesi)	Errore sconosciuto

Codici di errore estesi

Il software RSLogix 5000 non visualizza testo per i codici di errore estesi.

Questi sono i codici di errore estesi per il codice di errore **0001**.

Codice di errore esteso (esad.)	Descrizione
0100	Connessione in uso
0103	Trasporto non supportato
0106	Conflitto di proprietà
0107	Connessione non trovata
0108	Tipo di connessione non valida
0109	Dimensione connessione non valida
0110	Modulo non configurato
0111	EPR non supportato
0114	Modulo errato
0115	Tipo di dispositivo errato
0116	Revisione errata
0118	Formato di configurazione non valido
011A	Connessioni non disponibili per l'applicazione

Codice di errore esteso (esad.)	Descrizione
0203	Timeout connessione
0204	Timeout di messaggio non connesso
0205	Errore parametro di invio non connesso
0206	Messaggio troppo grande
0301	Memoria buffer insufficiente
0302	Larghezza di banda non disponibile
0303	Nessun filtro disponibile
0305	Corrispondenza firma
0311	Porta non disponibile
0312	Indirizzo connessione non disponibile
0315	Tipo di segmento non valido
0317	Connessione non pianificata

Questi sono i codici di errore estesi per il codice di errore **001F**.

Codice di errore esteso (esad.)	Descrizione
0203	Timeout connessione

Questi sono i codici di errore estesi per i codici di errore **0004** e **0005**.

Codice di errore esteso (esad.)	Descrizione
0000	Memoria insufficiente per lo stato esteso
0001	Istanze insufficienti per lo stato esteso

Questi sono i codici di errore estesi per il codice di errore **00FF**.

Codice di errore esteso (esad.)	Descrizione
2001	IOI eccessivo
2002	Valore parametro errato
2018	Rifiuto semaforo
201B	Dimensioni troppo piccole
201C	Dimensione non valida
2100	Errore di privilegio
2101	Posizione del selettore a chiave non valida
2102	Password non valida
2103	Nessuna password configurata
2104	Indirizzo fuori gamma
2105	Indirizzo e quantità fuori gamma
2106	Dati in uso
2107	Tipo non valido o non supportato

Codice di errore esteso (esad.)	Descrizione
2108	Controllore in modalità di caricamento o di scaricamento
2109	Si è tentato di modificare il numero di dimensioni della matrice
210A	Nome simbolo non valido
210B	Il simbolo non esiste
210E	Ricerca nulla
210F	La task non può iniziare
2110	Impossibile scrivere
2111	Impossibile leggere
2112	Routine condivisa non modificabile
2113	Controllore in modalità di errore
2114	Modalità di esecuzione inibita

Codici di errore di PLC e SLC (.ERR)

La versione firmware di Logix 10.x e successive forniscono nuovi codici di errore per gli errori associati ai tipo di messaggio PLC e SLC (messaggi PCCC).

- Questa modifica consente al software RSLogix 5000 di visualizzare una descrizione più significativa di molti tra gli errori. In passato il software non forniva alcuna descrizione per gli errori associati al codice di errore 00F0.
- Questa modifica rende inoltre i codici di errore più coerenti con gli errori indicati da altri controllori, quali i PLC-5.

La tabella seguente mostra le modifiche nei codici di errore dalla versione R9.x e precedenti rispetto alla R10.x e successive. In conseguenza di tale modifica, il membro .ERR restituisce un unico valore per ogni errore PCCC. .EXERR non è più necessario per tali errori.

Codici di errore di PLC e SLC (esad.)

Versione R9.x e precedenti		**Versione R10.x e successive**		**Descrizione**
.ERR	**.EXERR**	**.ERR**	**.EXERR**	
0010		1000		Comando o formato non valido dal processore locale
0020		2000		Il modulo di comunicazione non funziona
0030		3000		Nodo remoto mancante, scollegato o spento
0040		4000		Processore collegato ma in errore (hardware)
0050		5000		Numero di stazione errato
0060		6000		La funzione richiesta non è disponibile
0070		7000		Processore in modalità Programmazione
0080		8000		Il file di compatibilità del processore non esiste
0090		9000		Il nodo remoto non può inserire in memoria il comando
00B0		B000		Il processore sta eseguendo uno scaricamento e non è accessibile
00F0	0001	F001		Il processore ha convertito l'indirizzo in modo errato
00F0	0002	F002		Indirizzo incompleto
00F0	0003	F003		Indirizzo errato
00F0	0004	F004		Formato indirizzo non valido – simbolo non trovato
00F0	0005	F005		Formato indirizzo non valido – il simbolo non riporta alcun carattere o riporta un numero di caratteri superiore a quello massimo supportato dal dispositivo
00F0	0006	F006		Il file di indirizzo non esiste nel processore di destinazione
00F0	0007	F007		Il file di destinazione è troppo piccolo per il numero di parole richiesto
00F0	0008	F008		Impossibile completare la richiesta La situazione è cambiata durante il funzionamento multipacchetto
00F0	0009	F009		Dati o file troppo grandi Memoria non disponibile

Codici di errore di PLC e SLC (esad.) (Continua)

Versione R9.x e precedenti		Versione R10.x e successive		Descrizione
.ERR	.EXERR	.ERR	.EXERR	
00F0	000A	F00A		Il processore di destinazione non può inserire le informazioni richieste in pacchetti
00F0	000B	F00B		Errore di privilegio; accesso negato
00F0	000C	F00C		La funzione richiesta non è disponibile
00F0	000D	F00D		La richiesta è ridondante
00F0	000E	F00E		Il comando non può essere eseguito
00F0	000F	F00F		Overflow; overflow istogramma
00F0	0010	F010		Nessun accesso
00F0	0011	F011		Il tipo di dati richiesto non corrisponde ai dati disponibili
00F0	0012	F012		Parametri di comando non corretti
00F0	0013	F013		Riferimento di indirizzo presente nell'area cancellata
00F0	0014	F014		Errore di esecuzione del comando per cause sconosciute Overflow istogramma del PLC-3
00F0	0015	F015		Errore di conversione dati
00F0	0016	F016		Lo scanner non è disponibile per comunicare con un adattatore rack 1771
00F0	0017	F017		La scheda non è disponibile per comunicare con il modulo
00F0	0018	F018		La risposta del modulo 1771 non era valida
00F0	0019	F019		Etichetta duplicata
00F0	001A	F01A		Proprietario file attivo – il file è in uso
00F0	001B	F01B		Proprietario programma attivo – qualcuno sta eseguendo uno scaricamento o delle modifiche online
00F0	001C	F01C		Il file del disco è protetto contro la scrittura o non è accessibile (solo offline)
00F0	001D	F01D		Il file del disco è utilizzato da un'altra applicazione Aggiornamento non eseguito (solo offline)

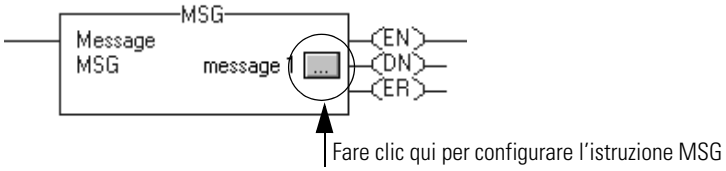
Codici di errore dei trasferimenti a blocchi

Questi sono i codici di errore specifici per i trasferimenti a blocchi in Logix5000.

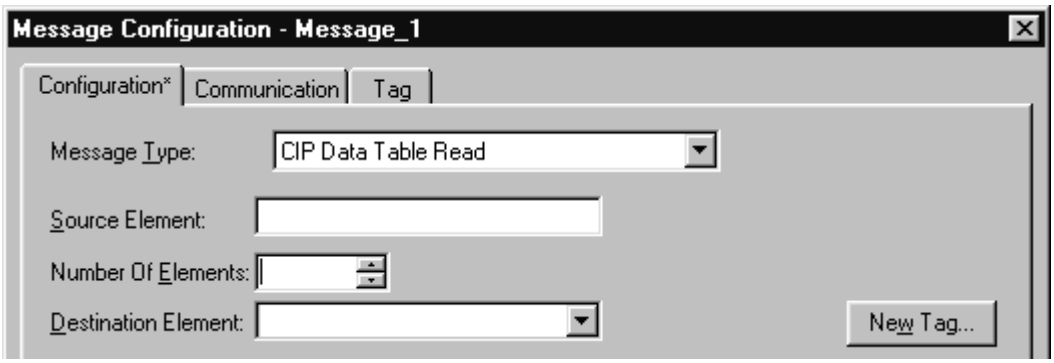
Codice di errore (esad.)	Descrizione	Visualizzazione nel software
00D0	Lo scanner non ha ricevuto una risposta di trasferimento a blocchi dal modulo a trasferimenti a blocchi entro 3,5 secondi dalla richiesta	Errore sconosciuto
00D1	Il checksum dalla risposta di lettura non corrisponde con il checksum del flusso dati	Errore sconosciuto
00D2	Lo scanner ha richiesto una lettura o una scrittura ma il modulo di trasferimento a blocchi ha risposto con l'opposto	Errore sconosciuto
00D3	Lo scanner ha richiesto una lunghezza ed il modulo di trasferimento a blocchi ha risposto con una lunghezza diversa	Errore sconosciuto
00D6	Lo scanner ha ricevuto una risposta dal modulo di trasferimento a blocchi indicante un errore nella richiesta di scrittura	Errore sconosciuto
00EA	Lo scanner non è stato configurato per comunicare con il rack contenente questo modulo di trasferimento a blocchi	Errore sconosciuto
00EB	Lo slot logico specificato non è disponibile per la dimensione del rack	Errore sconosciuto
00EC	Esiste una richiesta di trasferimento a blocchi in corso che richiede una risposta prima che possa avere inizio un'altra richiesta	Errore sconosciuto
00ED	La dimensione della richiesta di trasferimento a blocchi non è compatibile con le richieste di trasferimenti a blocchi di dimensioni valide	Errore sconosciuto
00EE	Il tipo di richiesta di trasferimento a blocchi non è compatibile con il previsto BT_READ o BT_WRITE	Errore sconosciuto
00EF	Lo scanner non è riuscito a trovare nella tabella dei trasferimenti a blocchi uno slot disponibile per la richiesta di trasferimento a blocchi	Errore sconosciuto
00F0	Lo scanner ha ricevuto la richiesta di ripristinare i canali I/O remoti in presenza di trasferimenti a blocchi in sospeso	Errore sconosciuto
00F3	Le code per i trasferimenti a blocchi remoti sono piene	Errore sconosciuto
00F5	Nessun canale di comunicazione configurato per il rack o lo slot richiesto	Errore sconosciuto
00F6	Nessun canale di comunicazione configurato per I/O remoto	Errore sconosciuto
00F7	Il timeout del trasferimento a blocchi, impostato nell'istruzione, è scaduto prima del completamento	Errore sconosciuto
00F8	Errore nel protocollo del trasferimento a blocchi – trasferimento a blocchi non richiesto	Errore sconosciuto
00F9	I dati del trasferimento a blocchi sono andati persi a causa di un canale di comunicazione guasto	Errore sconosciuto
00FA	Il modulo di trasferimento a blocchi ha richiesto una lunghezza diversa da quella associata all'istruzione di trasferimento a blocchi	Errore sconosciuto
00FB	Il checksum dei dati di trasferimento a blocchi di lettura era errato	Errore sconosciuto
00FC	Trasferimento non valido dei dati di trasferimento a blocchi di scrittura tra l'adattatore e il modulo a trasferimento a blocchi	Errore sconosciuto
00FD	La dimensione del trasferimento a blocchi più la dimensione dell'indice nella tabella dati dei trasferimenti a blocchi è maggiore della dimensione del file della tabella dati dei trasferimenti a blocchi	Errore sconosciuto

Specificare i dettagli di configurazione

Dopo avere immesso l'istruzione MSG e specificato la struttura MESSAGE, utilizzare la finestra di dialogo Message Configuration per specificare i dettagli del messaggio.



Le informazioni da configurare dipendono dal tipo di messaggio selezionato.



Se il dispositivo di destinazione è un	Selezionare uno di questi tipi di messaggio	Vedere pagina
Controllore Logix5000	Tabella dati CIP di lettura	157
	Tabella dati CIP di scrittura	
Modulo di I/O configurato con il software RSLogix 5000	Riconfigurazione modulo	158
	CIP generico	159
Controllore PLC-5	Lettura per PLC5	160
	Scrittura per PLC5	
	Lettura a parole per PLC5	
	Scrittura a parole per PLC5	
Controllore SLC Controllore MicroLogix	Lettura per SLC	162
	Scrittura per SLC	
Modulo di trasferimento a blocchi	Trasferimento a blocchi di lettura	162
	Trasferimento a blocchi di scrittura	
Processore PLC-3	Lettura per PLC3	163
	Scrittura per PLC3	
	Lettura a parole per PLC3	
	Scrittura a parole per PLC3	
Processore PLC-2	Lettura non protetta per PLC2	164
	Scrittura non protetta per PLC2	

È necessario specificare le seguenti informazioni di configurazione:

Per questa proprietà	Specificare
Source Element	• Se si seleziona un tipo di messaggio di lettura, l'elemento di origine è l'indirizzo dei dati che si desidera leggere nel dispositivo di destinazione. Utilizzare la sintassi di indirizzamento del dispositivo di destinazione. • Se si seleziona un tipo di messaggio di scrittura, il tag di origine è il primo elemento del tag che si desidera inviare al dispositivo di destinazione.
Number of Elements	Il numero di elementi che si leggono/scrivono dipende dal tipo di dati utilizzati. Un elemento si riferisce ad un "gruppo" di dati correlati. Ad esempio, il tag *timer1* è un elemento costituito da una struttura di controllo timer .
Destination Element	• Se si seleziona un tipo di messaggio di lettura, l'elemento di destinazione è il primo elemento del tag del controllore Logix5550 in cui si desidera memorizzare i dati letti dal dispositivo di destinazione. • Se si seleziona un tipo di messaggio di scrittura, l'elemento di destinazione è l'indirizzo del luogo del dispositivo di destinazione in cui si desidera scrivere i dati.

Specificare la Tabella dati CIP per messaggi di lettura e scrittura

I tipi di messaggio Tabella dati CIP di lettura e scrittura trasferiscono dati tra controllori Logix5000.

Selezionare questo comando	Se si desidera
Tabella dati CIP di lettura	leggere dati da un altro controllore. I tipi di Source e di Destination devono corrispondere.
Tabella dati CIP di scrittura	scrivere dati in un altro controllore. I tipi di Source e di Destination devono corrispondere.

Riconfigurare un modulo di I/O

Utilizzare il messaggio Riconfigurazione modulo per inviare nuove informazioni di configurazione a un modulo di I/O. Durante la riconfigurazione:

- I moduli di ingresso continuano a inviare dati di ingresso al controllore.
- I moduli di uscita continuano a controllare i propri dispositivi di uscita.

Un messaggio di Riconfigurazione modulo richiede le seguenti proprietà di configurazione:

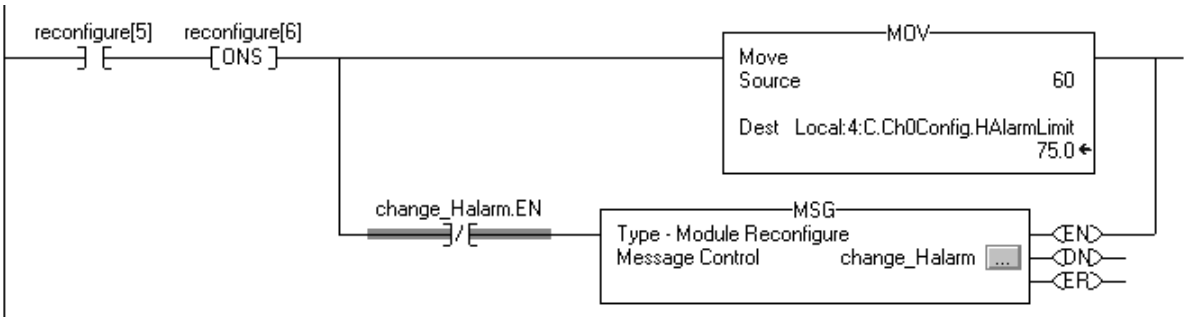
In questa proprietà	Selezionare
Message Type	Module Reconfigure

Esempio: Per riconfigurare un modulo di I/O:

1. Impostare il relativo membro del tag di configurazione del modulo al nuovo valore.
2. Inviare un messaggio di Riconfigurazione modulo al modulo.

Quando *reconfigure[5]* viene impostato, impostare l'allarme di livello massimo a 60 per il modulo locale nello slot 4. Il messaggio di Riconfigurazione modulo invierà il nuovo valore di allarme al modulo. L'istruzione one shot impedisce al ramo di inviare più messaggi al modulo mentre *reconfigure[5]* è attivo.

Ladder



Testo strutturato

```
IF reconfigure[5] AND NOT reconfigure[6]THEN
     Local:4:C.Ch0Config.HAlarmLimit := 60;
     IF NOT change_Halarm.EN THEN
          MSG(change_Halarm);
     END_IF;
END_IF;
     reconfigure[6] := reconfigure[5];
```

Specificare messaggi CIP generici

Un messaggio CIP generico esegue un'azione specifica su un modulo I/O.

Se si desidera	In questa proprietà	Digitare o selezionare	
Eseguire una prova a impulsi su un modulo di uscita digitale	Message Type	CIP generico	
	Service Type	Pulse Test	
	Source	*nome_tag* di tipo INT [5]	
		Questa matrice contiene:	
		nome_tag[0]	maschera a bit dei punti da testare (testare solo un punto alla volta)
		nome_tag[1]	riservato, lasciare 0
		nome_tag[2]	ampiezza di impulso (in centinaia di μsec, generalmente 20)
		nome_tag[3]	ritardo dell'attraversamento dello zero per l'I/O ControlLogix (in centinaia di μsec, generalmente 40)
		nome_tag[4]	verifica ritardo
	Destination	lasciare vuoto	
Ripristinare i fusibili elettronici di un modulo di uscita digitale	Message Type	CIP generico	
	Service Type	Reset Electronic Fuse	
	Source	*nome tag* di tipo DINT Questo tag rappresenta una maschera a bit dei punti su cui ripristinare i fusibili.	
	Destination	lasciare vuoto	
Azzerare la diagnostica agganciata di un modulo di ingresso digitale	Message Type	CIP generico	
	Service Type	Reset Latched Diagnostics (I)	
	Source	*nome_tag* di tipo DINT Questo tag rappresenta una maschera a bit dei punti su cui ripristinare la diagnostica.	
Azzerare la diagnostica agganciata di un modulo di uscita digitale	Message Type	CIP generico	
	Service Type	Reset Latched Diagnostics (O)	
	Source	*nome_tag* di tipo DINT Questo tag rappresenta una maschera a bit dei punti su cui ripristinare la diagnostica.	

Se si desidera	In questa proprietà	Digitare o selezionare
Sganciare l'allarme di un modulo di ingresso analogico	Message Type	CIP generico
	Service Type	Selezionare l'allarme da sganciare: • Unlatch All Alarms (I) • Unlatch Analog High Alarm (I) • Unlatch Analog High High Alarm (I) • Unlatch Analog Low Alarm (I) • Unlatch Analog Low Low Alarm (I) • Unlatch Rate Alarm (I)
	Instance	Canale dell'allarme da sganciare
Sganciare l'allarme di un modulo di uscita analogico	Message Type	CIP generico
	Service Type	Selezionare l'allarme da sganciare: • Unlatch All Alarms (O) • Unlatch High Alarm (O) • Unlatch Low Alarm (O) • Unlatch Ramp Alarm (O)
	Instance	Canale dell'allarme da sganciare

Specificare messaggi per PLC-5

Usare i tipi di messaggio PLC-5 per comunicare con controllori PLC-5.

Selezionare questo comando	Se si desidera
Lettura per PLC5	Leggere dati di tipo intero a 16 bit, a virgola mobile o stringa e conservare l'integrità dei dati. Vedere Tipi di dati per messaggi di lettura e scrittura per PLC5 a pagina 161.
Scrittura per PLC5	Scrivere dati di tipo intero a 16 bit, a virgola mobile o stringa e conservare l'integrità dei dati. Vedere Tipi di dati per messaggi di lettura e scrittura per PLC5 a pagina 161.
Lettura a parole per PLC5	Leggere una serie contigua di parole a 16 bit nella memoria del PLC-5 a prescindere dal tipo di dati. Questo comando inizia dall'indirizzo specificato come Elemento Source e legge in sequenza il numero delle parole a 16 bit richieste. I dati dall'Elemento Source vengono memorizzati, a partire dall'indirizzo specificato come tag di destinazione.
Scrittura a parole per PLC5	Scrivere parole contigue di 16 bit dalla memoria del Logix5000 alla memoria del PLC-5, a prescindere dal tipo di dati. Questo comando inizia dall'indirizzo specificato come Tag Source e legge in sequenza il numero delle parole a 16 bit richieste. I dati dall'Elemento Source vengono memorizzati, a partire dall'indirizzo specificato come tag di destinazione nel processore PLC-5.

La seguente tabella mostra i tipi di dati da usare con messaggi di lettura e scrittura per PLC5.

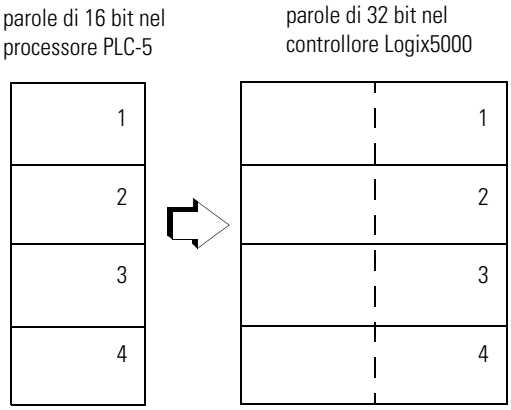
Tipi di dati per messaggi di lettura e scrittura per PLC5

Per il tipo dati PLC-5	Usare il tipo dati Logix5000
B	INT
F	REAL
N	INT
	DINT (scrivere valori DINT su un controllore PLC-5 solo se il valore è ≥ –32 768 e ≤ 32 767.)
S	INT
ST	STRING

I comandi di Lettura e di Scrittura per PLC5 funzionano anche con i processori SLC 5/03 (OS303 e superiori), i processori SLC 5/04 (OS402 e superiori) e i processori SLC 5/05.

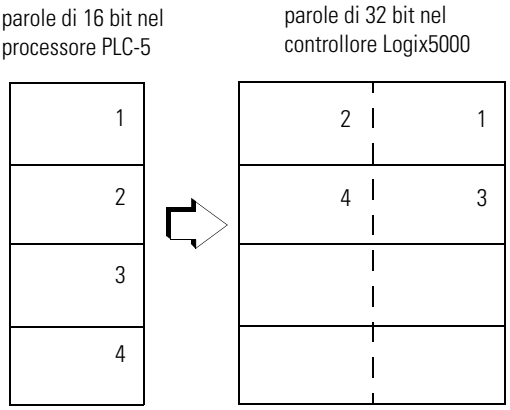
I seguenti schemi mostrano le differenze tra i comandi specializzati e a parole. L'esempio utilizza comandi di lettura da un processore PLC-5 ad un controllore Logix5000.

Comando di lettura specializzato



I comandi specializzati mantengono la struttura ed il valore dei dati.

Comando di lettura a parole



I comandi a parole occupano i tag di destinazione in modo contiguo. La struttura ed il valore dei dati cambiano a seconda del tipo di dati di destinazione.

Specificare messaggi per SLC

Usare i tipi di messaggio SLC per comunicare con controllori SLC e MicroLogix. La seguente tabella mostra i tipi di dati cui l'istruzione consente di accedere. La tabella mostra anche il tipo dati Logix5000 corrispondente.

Per il tipo dati SLC o MicroLogix	Usare il tipo dati Logix5000
F	REAL
L (controllori MicroLogix 1200 e 1500)	DINT
N	INT

Specificare messaggi di trasferimento a blocchi

I messaggi di trasferimento a blocchi sono usati per comunicare con moduli a trasferimento a blocchi su una rete I/O remoto universale.

Se si desidera	Selezionare questo comando
leggere dati da un modulo a trasferimento a blocchi. Questo tipo di messaggio sostituisce l'istruzione BTR.	Trasferimento a blocchi di lettura
scrivere dati ad un modulo a trasferimento a blocchi. Questo tipo di messaggio sostituisce l'istruzione BTW.	Trasferimento a blocchi di scrittura

Seguire queste linee guida per configurare un messaggio di trasferimento a blocchi:

- I tag di origine (per i BTW) e di destinazione (per i BTR) devono essere abbastanza grandi da accettare i dati richiesti, eccetto le strutture MESSAGE, AXIS e MODULE.
- Specificare quanti interi a 16 bit (INT) inviare o ricevere. È possibile specificare da 0 a 64 interi.

Se si desidera che	Specificare
il modulo di trasferimento a blocchi determini il numero di interi a 16 bit da inviare (BTR).	0 per il numero di elementi
il controllore invii 64 interi (BTW).	

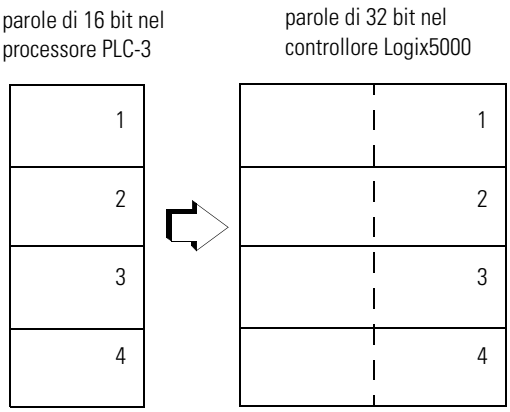
Specificare messaggi per PLC-3

I messaggi per PLC-3 sono rivolti ai processori PLC-3.

Selezionare questo comando	Se si desidera
Lettura per PLC3	leggere dati di tipo intero o REAL. Per i numeri interi, questo comando legge interi a 16 bit dal processore PLC-3 che memorizza in matrici di dati SINT, INT o DINT del controllore Logix5000, e gestisce l'integrità dei dati. Questo comando, inoltre, legge i dati a virgola mobile dal PLC-3 e li memorizza in un tag di tipo di dati REAL del controllore Logix5000.
Scrittura per PLC3	scrivere dati di tipo intero o REAL. Questo comando scrive dati SINT o INT nel file intero del PLC-3 e gestisce l'integrità dei dati. È possibile scrivere dati DINT purché essi rientrino nel tipo di dati INT (–32 768 ≥ dati ≤ 32 767). Questo comando, inoltre, scrive dati di tipo REAL dal controllore Logix5000 in un file a virgola mobile di PLC-3.
Lettura a parole per PLC3	leggere una serie contigua di parole a 16 bit nella memoria del PLC-3 a prescindere dal tipo di dati. Questo comando inizia dall'indirizzo specificato come Elemento Source e legge in sequenza il numero delle parole a 16 bit richieste. I dati dall'Elemento Source vengono memorizzati, a partire dall'indirizzo specificato come tag di destinazione.
Scrittura a parole per PLC3	scrivere parole contigue di 16 bit dalla memoria del Logix5000 alla memoria del PLC-3, a prescindere dal tipo di dati. Questo comando inizia dall'indirizzo specificato come Tag Source e legge in sequenza il numero delle parole a 16 bit richieste. I dati dall'Elemento Source vengono memorizzati a partire dall'indirizzo specificato come Elemento di destinazione nel processore PLC-3.

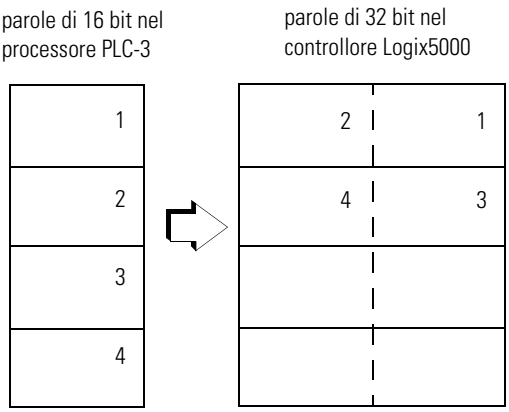
I seguenti schemi mostrano le differenze tra i comandi specializzati e a parole. L'esempio utilizza comandi di lettura da un processore PLC-3 ad un controllore Logix5000.

Comando di lettura specializzata



I comandi specializzati mantengono la struttura ed il valore dei dati.

Comando di lettura a parole



I comandi a parole occupano i tag di destinazione in modo contiguo. La struttura ed il valore dei dati cambiano a seconda del tipo di dati di destinazione.

Specificare messaggi per PLC-2

I messaggi per PLC-2 sono rivolti ai processori PLC-2.

Selezionare questo comando	Se si desidera
Lettura non protetta per PLC2	leggere parole a 16 bit da qualsiasi area della tabella dati del PLC-2 o dal file di compatibilità con il PLC-2 di un altro processore.
Scrittura non protetta per PLC2	scrivere parole a 16 bit in qualsiasi area della tabella dati del PLC-2 o dal file di compatibilità con il PLC-2 di un altro processore.

Questo messaggio di trasferimento utilizza parole a 16 bit, pertanto assicurarsi che il tag del Logix5000 memorizzi correttamente i dati trasferiti (in genere come una matrice INT).

Esempi di configurazione MSG:

I seguenti esempi mostrano tag ed elementi di origine e destinazione per varie combinazioni di controllore.

Per le istruzioni MSG che hanno origine da un controllore Logix5000 e che scrivono in un altro controllore:

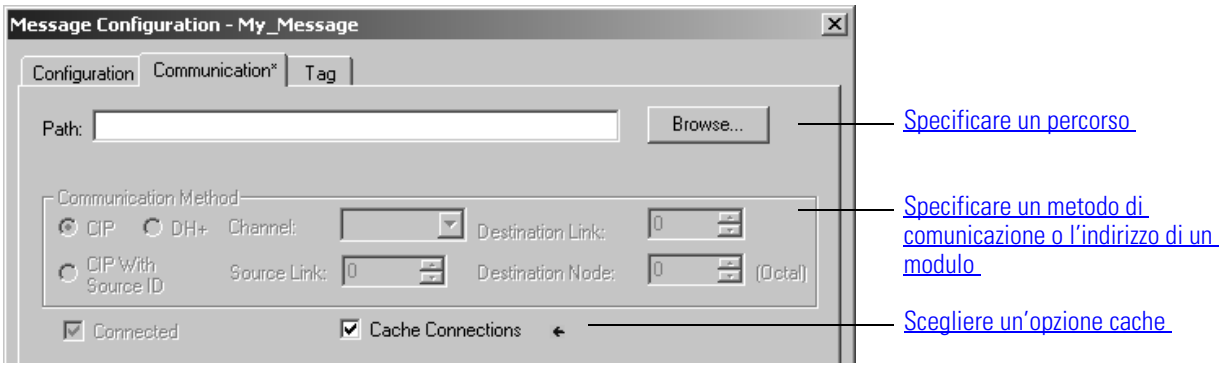
Percorso del messaggio	Origine e Destinazione di esempio	
Logix5000 → Logix5000	tag di origine	*matrice_1[0]*
	tag di destinazione	*matrice_2[0]*
	È possibile utilizzare un tag alias al posto del tag di origine (nel controllore Logix5000 di partenza).	
	Non è possibile utilizzare un alias al posto del tag di destinazione. La destinazione deve essere un tag base.	
Logix5000 → PLC-5 Logix5000 → SLC	tag di origine	*matrice_1[0]*
	elemento di destinazione	*N7:10*
	È possibile utilizzare un tag alias al posto del tag di origine (nel controllore Logix5000 di partenza).	
Logix5000 → PLC-2	tag di origine	*matrice_1[0]*
	elemento di destinazione	*010*

Per le istruzioni MSG che hanno origine da un controllore Logix5000 e che leggono da un altro controllore:

Percorso del messaggio:	Origine e Destinazione di esempio	
Logix5000 → Logix5000	tag di origine	*matrice_1[0]*
	tag di destinazione	*matrice_2[0]*
	Non è possibile utilizzare un tag alias al posto del tag di origine. L'origine deve essere un tag base.	
	È possibile utilizzare un tag alias al posto del tag di destinazione (nel controllore Logix5000 di partenza).	
Logix5000 → PLC-5 Logix5000 → SLC	elemento di origine	*N7:10*
	tag di destinazione	*matrice_1[0]*
	È possibile utilizzare un tag alias al posto del tag di destinazione (nel controllore Logix5000 di partenza).	
Logix5000 → PLC-2	elemento di origine	*010*
	tag di destinazione	*matrice_1[0]*

Specificare i dettagli di comunicazione

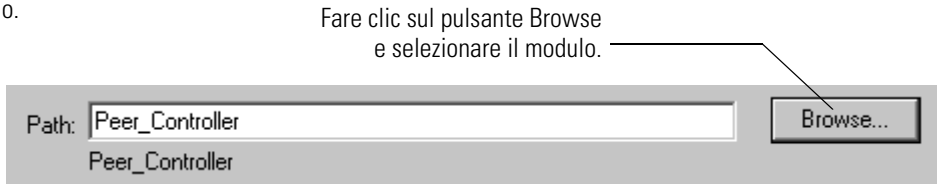
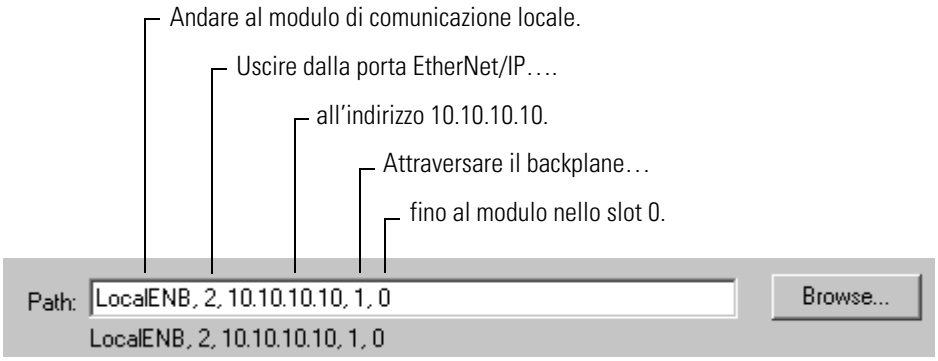
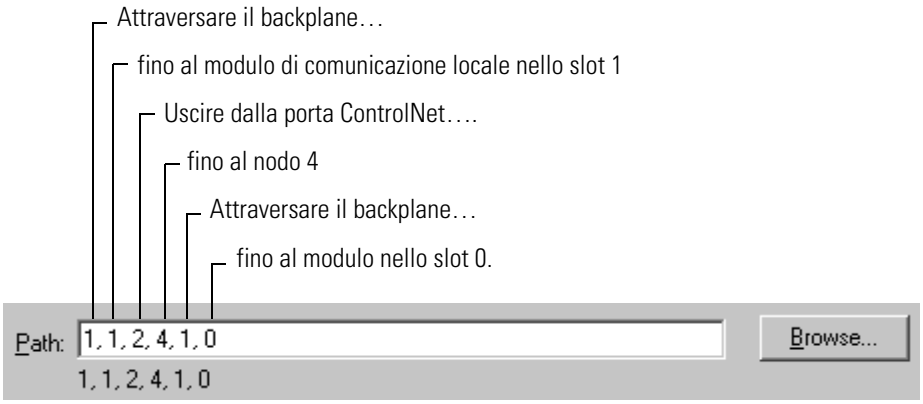
Quando si configura un'istruzione MSG, è necessario specificare le seguenti informazioni nella scheda Comunicazione.



Specificare un percorso

Il percorso descrive l'instradamento del messaggio per raggiungere la destinazione. Utilizza nomi dalla configurazione I/O del controllore, numeri digitati o entrambi.

Se	Allora
La configurazione I/O del controllore ha il modulo che riceve il messaggio.	Usare il pulsante *Browse* per selezionare il modulo di comunicazione locale.
La configurazione I/O del controllore ha solo il modulo di comunicazione locale.	1. Usare il pulsante *Browse* per selezionare il modulo di comunicazione locale. 2. Digitare il resto del percorso.
La configurazione I/O del controllore non ha nessuno dei moduli necessari per il messaggio.	Digitare il percorso.

Esempio	
La configurazione I/O del controllore ha il modulo che riceve il messaggio.	
La configurazione I/O del controllore ha solo il modulo di comunicazione locale.	
La configurazione I/O del controllore non ha nessuno dei moduli necessari per il messaggio.	

Per digitare un percorso, usare il seguente formato:

porta, indirizzo_successivo, porta, indirizzo_successivo, …

Dove	**È**	
	Per questa rete	**Tipo**
porta	backplane	1
	DF1 (seriale, canale seriale 0)	2
	ControlNet	
	EtherNet/IP	
	DH+ canale A	
	DH+ canale B	3
	DF1 canale 1 (canale seriale 1)	
indirizzo_successivo	backplane	numero di slot del modulo
	DF1 (seriale)	l'indirizzo della stazione (0–254)
	ControlNet	il numero di nodo (1–99 decimale)
	DH+	8# seguito dal numero di nodo (1–77 ottale) Ad esempio, per specificare l'indirizzo di nodo ottale di 37, digitare 8#37.
	EtherNet/IP	È possibile specificare un modulo su una rete Ethernet/IP utilizzando uno di questi formati: Indirizzo IP (ad esempio, 10.10.10.10) Indirizzo IP:Porta (ad esempio, 10.10.10.10:24) Nome DNS (ad esempio, tanks) Nome DNS:Porta (ad esempio, tanks:24)

Per trasferimenti a blocchi

Per messaggi di trasferimento a blocchi, aggiungere i seguenti moduli alla configurazione I/O del controllore:

Per trasferimenti a blocchi sulla rete	Aggiungere questi moduli alla configurazione I/O
ControlNet	• modulo di comunicazione locale (ad es., modulo 1756-CNB) • modulo adattatore remoto (ad es., modulo 1771-ACN)
Universal Remote I/O	• modulo di comunicazione locale (ad es., modulo 1756-DHRIO) • un modulo adattatore remoto (ad es., modulo 1771-ASB) per ogni rack, o parte di rack, dello chassis • modulo di trasferimento a blocchi (facoltativo)

Specificare un metodo di comunicazione o l'indirizzo di un modulo

Usare la seguente tabella per scegliere un metodo di comunicazione o l'indirizzo di un modulo per il messaggio.

Se il dispositivo di destinazione è un	Selezionare	E specificare	
Controllore Logix5000	**CIP**	non sono necessarie altre informazioni	
Controllore PLC-5 su una rete EtherNet/IP			
Controllore PLC-5 su una rete ControlNet			
Controllore SLC 5/05			
Controllore PLC-5 su una rete DH+	**DH+**	Canale:	Canale A o B del modulo 1756-DHRIO collegato alla rete DH+
Controllore SLC su una rete DH+		Collegamento di origine:	ID del collegamento assegnato al backplane del Controllore nella tabella di instradamento del modulo 1756-DHRIO. (Il nodo di origine della tabella di instradamento è automaticamente il numero di slot del Controllore.)
Processore PLC-3		Collegamento di destinazione	ID del collegamento DH+ remoto dove risiede il dispositivo di destinazione
Processore PLC-2		Nodo di destinazione:	Indirizzo della stazione del dispositivo di destinazione; ottale.
		Se c'è un solo collegamento DH+ e non è stato utilizzato il software RSLinx per configurare il modulo DH/RIO per collegamenti remoti, specificare 0 sia per il Collegamento di origine che per il Collegamento di destinazione.	

Se il dispositivo di destinazione è un	Selezionare	E specificare	
Applicazione su una workstation che riceve un messaggio non richiesto instradato su una rete Ethernet/IP o una rete ControlNet tramite RSLinx.	**CIP con ID origine** (Ciò consente all'applicazione di ricevere i dati da un Controllore.)	Collegamento di origine:	ID remoto dell'argomento nel software RSLinx
		Collegamento di destinazione:	Impostazione dell'ID di collegamento virtuale in RSLinx (0-65535)
		Nodo di destinazione:	ID di destinazione (0–77 ottale) fornito dall'applicazione ad RSLinx Come argomento DDE in RSLinx, usare 77.
		Il numero di slot del controllore ControlLogix viene utilizzato come Nodo di origine.	
modulo di trasferimento a blocchi su una rete I/O remoto universale	RIO	Canale:	Canale A o B del modulo 1756-DHRIO collegato alla rete RIO
		Rack	Numero di rack (ottale) del modulo
		Gruppo	Numero di gruppo del modulo
		Slot	Numero dello slot in cui si trova il modulo
modulo di trasferimento a blocchi su una rete ControlNet	ControlNet	Slot	Numero dello slot in cui si trova il modulo

Scegliere un'opzione cache

A seconda di come si configura un'istruzione MSG, può utilizzare una connessione per inviare o ricevere dati.

Tipo di messaggio	Metodo di comunicazione	Connessione
Lettura o scrittura tabella dati CIP	→	✓
PLC2, PLC3, PLC5, o SLC (tutti i tipi)	CIP CIP con ID origine	
	DH+	✓
CIP generico	→	a scelta [1]
Trasferimento a blocchi di lettura o scrittura	→	✓

[1] È possibile connettere i messaggi CIP generici. Per la maggior parte delle applicazioni, tuttavia, si consiglia di lasciare i messaggi CIP generici non connessi.

Se un'istruzione MSG usa una connessione, si ha la possibilità di lasciare aperta la connessione (cache) o chiuderla quando la trasmissione del messaggio è terminata.

Se	Allora
Si mette in cache la connessione	La connessione resta aperta al termine dell'istruzione MSG. Questo ottimizza i tempi di esecuzione. L'apertura di una connessione ogni volta che si deve eseguire un messaggio incrementa il tempo di esecuzione.
Non si mette in cache la connessione	La connessione viene chiusa al termine dell'istruzione MSG. In questo modo la connessione viene liberata per altri usi.

Il controllore ha i seguenti limiti in termini di numero di connessioni che è possibile mettere in cache:

Se si possiede questo software e versione firmware	Allora è possibile mettere in cache
11.x o precedenti	• messaggi di trasferimento a blocchi fino a 16 connessioni • altri tipi di messaggi fino a 16 connessioni
12.x o successive	fino a 32 connessioni

Se più messaggi vanno allo stesso dispositivo, i messaggi possono essere in grado di condividere una connessione.

Se le istruzioni MSG sono dirette a	E sono	Allora
diversi dispositivi	→	Ogni istruzione MSG usa 1 connessione.
lo stesso dispositivo	attivati contemporaneamente	Ogni istruzione MSG usa 1 connessione.
	NON attivati contemporaneamente	Le istruzioni MSG condividono la connessione (insieme contano come 1 connessione).

ESEMPIO Condivisone di una connessione

Se il controllore alterna tra l'invio di un messaggio di lettura a trasferimento di blocchi e un messaggio di scrittura a trasferimento di blocchi, insieme entrambi i messaggi contano come 1 connessione. Mettere in cache entrambi i messaggi conta come 1 nell'elenco della cache.

Regole generali

Quando si pianificano e si programmano le istruzioni MSG, seguire queste regole generali

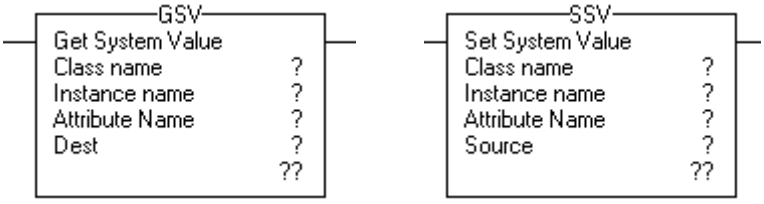
Regola	Dettagli
1. Per ogni istruzione MSG, creare un tag di controllo.	Ogni istruzione MSG richiede il proprio tag di controllo. • Tipo di dati = MESSAGE • Ambito = controllore • Il tag *non può* far parte di una matrice né di un tipo di dati definito dall'utente.
2. Mantenere i dati di sorgente e/o destinazione nell'ambito del controllore.	Un'istruzione MSG è in grado di accedere solo ai tag che si trovano nella cartella dei tag del controllore (ambito del controllore).
3. Se il MSG è diretto a un dispositivo che usa numeri interi a 16 bit, usare un buffer di INT nei MSG e DINT di tutto il progetto.	Se il messaggio è diretto a un dispositivo che usa numeri interi a 16 bit, come un controllore PLC-5® o SLC 500™ e trasferisce numeri interi (non REAL), usare un buffer di INT nel messaggio e DINT in tutto il progetto. Questo aumenta l'efficienza del progetto perché i controllori Logix lavorano con maggior efficienza e usano meno memoria quando lavorano con numeri interi a 32 bit (DINT). Per passare da INT a DINT, vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.
4. Mettere in cache i messaggi connessi che vengono eseguiti più frequentemente.	Mettere in cache la connessione per le istruzioni MSG che vengono eseguite più frequentemente, fino al numero massimo consentito per la propria versione del controllore. Questo ottimizza i tempi di esecuzione perché il controllore non deve aprire una connessione ogni volta che il messaggio viene eseguito.
5. Per attivare più di 16 MSG per volta, usare una qualche strategia di gestione.	Se vengono attivati più di 16 MSG per volta, alcune istruzioni MSG potrebbero subire dei ritardi nell'entrare in coda. Per garantire l'esecuzione di ogni messaggio, usare una delle seguenti opzioni: • Attivare ogni messaggio in sequenza. • Attivare i messaggi in gruppi. • Programmare un messaggio affinché comunichi con più dispositivi. Per ulteriori informazioni, vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001. • Programmare la logica affinché coordini l'esecuzione dei messaggi. Per ulteriori informazioni, vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.
6. Mantenere il numero di MSG non connessi e non in cache inferiore del numero di buffer non connessi.	Il controllore può avere 10–40 buffer non connessi. Il valore di default è 10. • Se tutti i buffer non connessi sono utilizzati quando un'istruzione lascia la coda dei messaggi, l'istruzione subisce un errore e non trasferisce i dati. • È possibile aumentare il numero di buffer non connessi (max. 40) ma continuare a seguire la regola 5. • Per aumentare il numero di buffer non connessi, vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.

Get System Value (GSV) and Set System Value (SSV)

Le istruzioni GSV/SSV servono per ottenere ed impostare i dati di sistema del controllore memorizzati negli oggetti.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Class name		nome	nome dell'oggetto
Instance name		nome	nome dell'oggetto specifico, quando l'oggetto richiede il nome
Attribute Name		nome	attributo dell'oggetto il tipo di dati dipende dall'attributo selezionato
Destination (GSV)	SINT INT DINT REAL struttura	tag	destinazione dei dati dell'attributo
Source (SSV)	SINT INT DINT REAL struttura	tag	tag contenente i dati che si desidera copiare nell'attributo

Testo strutturato

```
GSV(ClassName,InstanceName,AttributeName,Dest);
SSV(ClassName,InstanceName,AttributeName,Source);
```

Gli operandi sono uguali a quelli usati per le istruzioni GSV e SSV ladder.

Descrizione: Le istruzioni GSV/SSV servono per ottenere ed impostare i dati di sistema del controllore memorizzati negli oggetti. Il controllore

memorizza i dati di sistema in oggetti. Non c'è un file di stato come nel processore PLC-5.

Quando è abilitata, l'istruzione GSV recupera le informazioni specifiche e le inserisce nella destinazione. Quando è abilitata, l'istruzione SSV imposta l'attributo specifico con i dati dell'origine.

Quando si immette un'istruzione GSV/SSV, il software di programmazione visualizza le classi di oggetto valide, i nomi degli oggetti e i nomi degli attributi di ciascuna istruzione. Per l'istruzione GSV è possibile ottenere i valori per tutti gli attributi disponibili. Nel caso dell'istruzione SSV, il software visualizza solo quegli attributi che è possibile impostare (SSV).

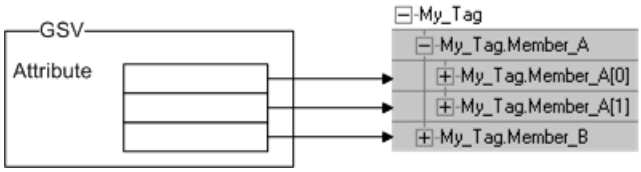
ATTENZIONE

Utilizzare le istruzioni GSV/SSV con cautela. Apportare modifiche agli oggetti può causare un funzionamento imprevisto del controllore o rischi per le persone.

È ***necessario*** testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

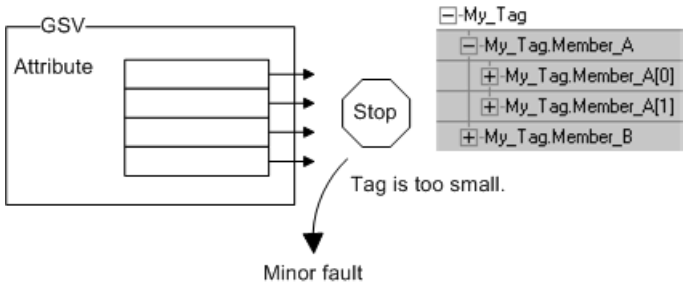
Le istruzioni GSV e SSV scrivono o leggono da un membro all' altro di un tag. Se il tag è troppo piccolo, le istruzioni non scrivono né leggono i dati ma registrano un errore minore.

Esempio 1



Il membro_A è troppo piccolo per l'attributo. L'istruzione GSV scrive l'ultimo valore nel membro_B.

Esempio 2



My_Tag è troppo piccolo per l'attributo. L'istruzione GSV si ferma e registra un errore minore.

Il paragrafo Oggetti GSV/SSV definisce gli attributi di ciascun oggetto ed i relativi tipi di dati. Ad esempio, l'attributo MajorFaultRecord dell'oggetto Program richiede un tipo di dati DINT[11].

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

Si verifica un errore minore se	Tipo errore	Codice errore
l'indirizzo dell'oggetto non è valido	4	5
si è specificato un oggetto che non supporta GSV/SSV	4	6
l'attributo non è valido	4	6
non si sono fornite informazioni sufficienti per un'istruzione SSV	4	6
la destinazione GSV non è sufficientemente grande da mantenere i dati richiesti	4	7

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
l'istruzione viene eseguita	Richiede o imposta il valore specificato.	Richiede o imposta il valore specificato.
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Oggetti GSV/SSV

Quando si immette un'istruzione GSV/SSV, si specifica l'oggetto e l'attributo a cui si vuole accedere. In alcuni casi vi sarà più di una presenza dello stesso tipo di oggetto, per cui è necessario specificare anche il nome dell'oggetto. Per esempio vi possono essere vari task nell'applicazione. Ogni task ha il proprio oggetto TASK a cui si può accedere con il nome del task.

ATTENZIONE

Nel caso dell'istruzione GSV, solo la dimensione specificata di dati viene copiata nella destinazione. Se, per esempio, l'attributo viene definito come SINT e la destinazione è un DINT, solo gli 8 bit meno significativi della destinazione DINT verranno aggiornati mentre i rimanenti 24 bit resteranno immutati.

È possibile accedere ai seguenti oggetti:

Per informazioni su questo oggetto	Consultare questa pagina o pubblicazione
AXIS	*ControlLogix Motion Module Setup and Configuration Manual*, pubblicazione 1756-UM006
CONTROLLER	177
CONTROLLERDEVICE	178
CST	179
DF1	181
FAULTLOG	184
MESSAGE	185
MODULE	187
MOTIONGROUP	188
PROGRAM	189
ROUTINE	190
SERIALPORT	190
TASK	192
WALLCLOCKTIME	194

Accesso all'oggetto CONTROLLER

L'oggetto CONTROLLER fornisce informazioni di stato riguardo l'esecuzione di un controllore.

Attributo	Tipo di dati	Istruzione	Descrizione
TimeSlice	INT	GSV SSV	Percentuale di CPU disponibile assegnata alla comunicazione. Valori validi sono 10–90. Questo valore non può essere modificato quando il selettore a chiave del controllore è in posizione esecuzione.
ControllerLogTotalEntry Count	DINT	SSV GSV	Numero di voci del registro del controllore dall'ultimo aggiornamento del firmware. Il numero viene azzerato se la RAM assume uno stato di errore. Il numero ha come limite il DINT più elevato.
ControllerLogExecution ModificationCount	DINT	SSV GSV	Numero di voci del registro del controllore che hanno avuto origine da una modifica delle proprietà di un programma/task, una modifica online o una modifica del timeslice del controllore. Può inoltre essere configurato in modo da comprendere le voci di registro che hanno origine dalle forzature. Il numero viene azzerato se la RAM assume uno stato di errore. Il numero non ha come limite il DINT maggiore e può verificarsi un rollover del contatore.
ControllerLogUnsavedEntry Count	DINT	GSV	Numero di voci nel registro del controllore che devono ancora essere memorizzate in un supporto rimovibile. Va da 0 al numero massimo di voci.
ControllerLogAutoWrite	BOOL	MSG	Flag usato per determinare se la scrittura automatica del registro del controllore sul supporto rimovibile è abilitata. 0 = la scrittura automatica è disabilitata (errore) 1 = il registro del controllore tenta di scrivere sul supporto rimovibile quando il registro è pieno all'80 %.
ExecutionCountConfigure Mask	DINT	MSG	Matrice di bit usata per determinare cosa causa l'incremento di Modify Execution Count. 0 = default (tutto tranne le forzature) 1 = forzature comprese (tutto + forzature).

Accesso all'oggetto CONTROLLERDEVICE

L'oggetto CONTROLLERDEVICE identifica l'hardware fisico del controllore.

Attributo	Tipo di dati	Istruzione	Descrizione
DeviceName	SINT[33]	GSV	La stringa ASCII che identifica il numero del catalogo del Controllore e della scheda di memoria. Il primo byte contiene un conteggio del numero di caratteri ASCII riportati nella stringa matrice.
ProductCode	INT	GSV	Identifica il tipo di controllore
ProductRev	INT	GSV	Indica la revisione corrente del prodotto. Il display deve essere esadecimale. Il byte basso contiene la revisione principale; il byte alto contiene la revisione secondaria.
SerialNumber	DINT	GSV	Numero seriale del dispositivo. Il numero seriale viene assegnato al dispositivo al momento della sua costruzione.

Controllore Logix	Codice prodotto
CompactLogix5320	43
CompactLogix5330	44
CompactLogix5335E	65
ControlLogix5550	3
ControlLogix5553	50
ControlLogix5555	51
ControlLogix5561	54
ControlLogix5562	55
ControlLogix5563	56
DriveLogix5720	48
FlexLogix5433	41
FlexLogix5434	42
SoftLogix5860	15

Attributo	Tipo di dati	Istruzione	Descrizione
Status	INT	GSV	Bit che indicano lo stato: I bit 3–0 sono riservati **Bit di stato del dispositivo** **Bit 7–4:** **Significato:** 0000 riservato 0001 aggiornamento memoria flash in corso 0010 riservato 0011 riservato 0100 la memoria flash è difettosa 0101 in errore 0110 esecuzione 0111 programmazione **Bit di stato di errore** **Bit 11–8:** **Significato:** 0001 errore minore recuperabile 0010 errore minore non recuperabile 0100 errore grave recuperabile 1000 errore grave non recuperabile **Bit di stato specifici del Logix5000** **Bit 13–12:** **Significato:** 01 selettore a chiave in posizione esecuzione 10 selettore a chiave in posizione programmazione 11 selettore a chiave in posizione remoto **Bit 15–14** **Significato** 01 il controllore sta cambiando modalità 10 modalità debug se il controllore è in modalità esecuzione
Type	INT	GSV	Identifica il dispositivo come controllore. Controllore = 14
Vendor	INT	GSV	Identifica il produttore del dispositivo. Allen-Bradley = 0001

Accesso all'oggetto CST

L'oggetto CST (tempo di sistema coordinato) fornisce il tempo di sistema coordinato dei dispositivi di uno chassis.

Attributo	Tipo di dati	Istruzione	Descrizione	
CurrentStatus	INT	GSV	Stato corrente del tempo di sistema coordinato. I bit indicano:	
			Bit:	Significato:
			0	errore hardware del timer: l'hardware del timer interno del dispositivo è in stato di errore
			1	rampa abilitata: il valore corrente dei bit 16+ meno significativi del timer sale fino al valore richiesto invece di passare al valore più basso. Questi bit vengono manipolati tramite il metodo di sincronizzazione specifico della rete
			2	tempo di sistema master: l'oggetto CST è una sorgente di tempo master nel sistema ControlLogix
			3	sincronizzato: il CurrentValue a 64 bit dell'oggetto CST è sincronizzato da un oggetto CST master tramite un aggiornamento tempo di sistema
			4	rete locale master: l'oggetto CST è la sorgente di tempo master della rete locale
			5	in modalità relè: l'oggetto CST sta funzionando in modalità relè a tempo
			6	rilevato master duplicato: è stato rilevato un master di tempo della rete locale duplicato. Questo bit è sempre 0 per i nodi a tempo.
			7	non usato
			8–9	00 = nodo a tempo dipendente 01 = nodo a tempo master 10 = nodo a relè a tempo 11 = non usato
			10–15	non usato
CurrentValue	DINT[2]	GSV	Valore corrente del timer. DINT[0] contiene i 32 bit più bassi; DINT[1] contiene i 32 bit più alti. La sorgente del timer viene regolata in modo da combaciare con il valore fornito nei servizi di aggiornamento e tramite la sincronizzazione della rete di comunicazione locale. La regolazione viene fatta o salendo fino al valore richiesto o con un'impostazione immediata sul valore richiesto, così come riportato nell'attributo CurrentStatus.	

Accesso all'oggetto DF1

L'oggetto DF1 fornisce un'interfaccia verso il driver di comunicazione DF1 che è possibile configurare per la porta seriale.

Attributo	Tipo di dati	Istruzione	Descrizione
ACKTimeout	DINT	GSV	Il tempo entro cui ricevere un riscontro della trasmissione di un messaggio (solo punto a punto e master). valore valido 0–32 767. Ritardo in conteggi di 20 msec. Il valore predefinito è 50 (1 secondo).
DiagnosticCounters	INT[19]	GSV	Matrice di contatori diagnostici per il driver di comunicazione DF1.

offset parola	DF1 punto a punto	DF1 slave	master
0	firma (0x0043)	firma (0x0042)	firma (0x0044)
1	bit modem	bit modem	bit modem
2	pacchetti inviati	pacchetti inviati	pacchetti inviati
3	pacchetti ricevuti	pacchetti ricevuti	pacchetti ricevuti
4	pacchetti non consegnati	pacchetti non consegnati	pacchetti non consegnati
5	non usato	messaggi re-inviati	messaggi re-inviati
6	NAK ricevuti	NAK ricevuti	non usato
7	ENQ ricevuti	pacchetti di interrogazione richiesti	non usato
8	pacchetti errati con NAK	pacchetti errati senza ACK	pacchetti errati senza ACK
9	NAK inviati senza memoria	senza memoria senza ACK	non usato
10	pacchetti duplicati ricevuti	pacchetti duplicati ricevuti	pacchetti duplicati ricevuti
11	caratteri errati ricevuti	non usato	non usato
12	conteggio recuperi DCD	conteggio recuperi DCD	conteggio recuperi DCD
13	conteggio interruzioni modem	conteggio interruzioni modem	conteggio interruzioni modem
14	non usato	non usato	massimo tempo di scansione priorità
15	non usato	non usato	ultimo tempo di scansione priorità
16	non usato	non usato	massimo tempo di scansione normale
17	non usato	non usato	ultimo tempo di scansione normale
18	ENQ inviati	non usato	non usato

DuplicateDetection	SINT	GSV	Abilita il rilevamento di messaggi duplicati. **Valore:** **Significato:** 0 rilevamento messaggi duplicati disabilitato non zero rilevamento messaggi duplicati disabilitato
EmbeddedResponseEnable	SINT	GSV	Abilita la funzione di risposta integrata (solo punto a punto). **Valore:** **Significato:** 0 iniziato solo dopo averne ricevuto una (default) 1 abilitato incondizionatamente
ENQTransmitLimit	SINT	GSV	Il numero di richieste (ENQ) da inviare dopo un timeout di ACK (solo punto a punto). Valori validi sono 0–127. Il valore di default è 3.
EOTSuppression	SINT	GSV	Abilita la soppressione delle trasmissioni EOT in risposta a tutti i pacchetti di interrogazione (solo slave). **Valore:** **Significato:** 0 soppressione EOT disabilitata (disabilitata) non zero soppressione EOT abilitata

Attributo	Tipo di dati	Istruzione	Descrizione
ErrorDetection	SINT	GSV	Indica lo schema di rilevamento errori. **Valore:** **Significato:** 0 BCC (default) 1 CRC
MasterMessageTransmit	SINT	GSV	Il valore corrente della trasmissione messaggi master (solo master). **Valore:** **Significato:** 0 tra interrogazioni delle stazioni 1 in sequenza alle interrogazioni (al posto del numero di stazione del master) Il valore di default è 0.
NAKReceiveLimit	SINT	GSV	Il numero di NAK ricevuti in risposta ad un messaggio prima di chiudere la trasmissione (solo comunicazione punto a punto). Valori validi sono 0–127. Il valore di default è 3.
NormalPollGroupSize	INT	GSV	Il numero di stazioni da interrogare nella matrice dei nodi ad interrogazione normale dopo l'interrogazione di tutte le stazioni nella matrice dei nodi ad interrogazione con priorità (solo master). Valori validi sono 0–255. Il valore di default è 0.
PollingMode	SINT	GSV	Modalità di interrogazione corrente (solo master). **Valore:** **Significato:** 0 basata su messaggi, ma non permette agli slave di iniziare messaggi 1 basata su messaggi, ma permette agli slave di iniziare messaggi (default) 2 trasferimento standard, con messaggio singolo per scansione di nodo 3 trasferimento standard, con più messaggi per scansione di nodo Il valore di default è 1.
ReplyMessageWait	DINT	GSV	Il tempo (mentre si comporta da master) che bisogna attendere dopo il ricevimento di un ACK prima di interrogare lo slave per una risposta (solo slave). Valore valido 0–65 535. Ritardo in unità da 20 msec. Il valore di default è 5 unità (100 msec).
StationAddress	INT	GSV	L'indirizzo della stazione corrente della porta seriale. Valori validi sono 0–254. Il valore di default è 0.
SlavePollTimeout	DINT	GSV	Il tempo espresso in msec. che lo slave attende affinché il master esegua l'interrogazione prima che lo slave dichiari che è impossibilitato a trasmettere poiché il master è inattivo (solo slave). valore valido 0–32 767. Ritardo in conteggi di 20 msec. Il valore di default è 3000 unità (1 minuto).
TransmitRetries	SINT	GSV	Il numero di tentativi di invio di un messaggio senza ottenere un riconoscimento (solo master e slave). Valori validi sono 0–127. Il valore di default è 3.

Attributo	Tipo di dati	Istruzione	Descrizione
PendingACKTimeout	DINT	SSV	Valore in attesa dell'attributo ACKTimeout.
PendingDuplicateDetection	SINT	SSV	Valore in attesa dell'attributo DuplicateDetection.
PendingEmbeddedResponse Enable	SINT	SSV	Valore in attesa dell'attributo EmbeddedResponse.
PendingENQTransmitLimit	SINT	SSV	Valore in attesa dell'attributo ENQTransmitLimit.
PendingEOTSuppression	SINT	SSV	Valore in attesa dell'attributo EOTSuppression.
PendingErrorDetection	SINT	SSV	Valore in attesa dell'attributo ErrorDetection.
PendingNormalPollGroupSize	INT	SSV	Valore in attesa per l'attributo NormalPollGroupSize.
PendingMasterMessage Transmit	SINT	SSV	Valore in attesa per l'attributo MasterMessageTransmit.
PendingNAKReceiveLimit	SINT	SSV	Valore in attesa per l'attributo NAKReceiveLimit.
PendingPollingMode	SINT	SSV	Valore in attesa per l'attributo PollingMode.
PendingReplyMessageWait	DINT	SSV	Valore in attesa per l'attributo ReplyMessageWait.
PendingStationAddress	INT	SSV	Valore in attesa per l'attributo StationAddress.
PendingSlavePollTimeout	DINT	SSV	Valore in attesa per l'attributo SlavePollTimeout.
PendingTransmitRetries	SINT	SSV	Valore in attesa per l'attributo TransmitRetries.

Per applicare i valori per ciascuno degli attributi di DF1 in attesa:

1. Utilizzare un'istruzione SSV per impostare il valore per l'attributo in attesa.

   È possibile impostare tutti gli attributi in attesa che si desidera utilizzando un'istruzione SSV per ogni attributo in attesa.

2. Utilizzare un'istruzione MSG per applicare il valore. L'istruzione MSG si applica ad ogni attributo in attesa che si è impostato. Configurare l'istruzione MSG come segue:

Scheda di configurazione MSG	Campo	Valore
Configuration	Message Type	CIP generico
	Service Code	0d esad.
	Object Type	a2
	Object ID	1
	Object attribute	lasciare vuoto
	Source	lasciare vuoto
	Number of Elements	0
	Destination	lasciare vuoto
Comunication	Path	percorso di comunicazione verso se stesso (1,*s* dove *s* = numero di slot del controllore)

Accesso all'oggetto FAULTLOG

L'oggetto FAULTLOG fornisce informazioni sugli errori del controllore.

Attributo	Tipo di dati	Istruzione	Descrizione
MajorEvents	INT	GSV SSV	Il numero di errori gravi che si sono verificati dall'ultima volta che questo contatore è stato azzerato.
MinorEvents	INT	GSV SSV	Il numero di errori minori che si sono verificati dall'ultima volta che questo contatore è stato azzerato.
MajorFaultBits	DINT	GSV SSV	I singoli bit indicano il motivo dell'errore grave corrente. **Bit: Significato:** 1 perdita di alimentazione 3 I/O 4 esecuzione dell'istruzione (programma) 5 programma di gestione errori 6 watchdog 7 stack 8 cambiamento di modalità 11 movimento
MinorFaultBits	DINT	GSV SSV	I singoli bit indicano il motivo dell'errore minore corrente. **Bit: Significato:** 4 esecuzione dell'istruzione (programma) 6 watchdog 9 porta seriale 10 batteria

Accesso all'oggetto MESSAGE

È possibile accedere all'oggetto MESSAGE mediante le istruzioni GSV/SSV. Specificare il nome del tag del messaggio per determinare l'oggetto MESSAGE desiderato. L'oggetto MESSAGE fornisce un'interfaccia per impostare ed attivare la comunicazione peer-to-peer. Questo oggetto sostituisce il tipo di dati MG del processore PLC-5.

Attributo	Tipo di dati	Istruzione	Descrizione
ConnectionPath	SINT[130]	GSV SSV	Dati per l'impostazione del percorso di connessione. I primi due byte (byte basso e byte alto) rappresentano la lunghezza in byte del percorso di connessione.
ConnectionRate	DINT	GSV SSV	Frequenza pacchetto richiesto della connessione.
MessageType	SINT	GSV SSV	Specifica il tipo di messaggio. **Valore:** **Significato:** 0 non inizializzato
Port	SINT	GSV SSV	Indica su quale porta deve essere inviato il messaggio. **Valore:** **Significato:** 1 backplane 2 porta seriale
TimeoutMultiplier	SINT	GSV SSV	Determina quando una connessione deve essere considerata scaduta e chiusa. **Valore:** **Significato:** 0 la connessione scadrà in un tempo pari a 4 volte la frequenza di aggiornamento (default) 1 la connessione scadrà in un tempo pari ad 8 volte la frequenza di aggiornamento 2 la connessione scadrà in un tempo pari ad 16 volte la frequenza di aggiornamento
UnconnectedTimeout	DINT	GSV SSV	Il periodo di timeout è espresso in microsecondi per tutti i messaggi non connessi. Il valore di default è 30 000 000 microsecondi (30 secondi).

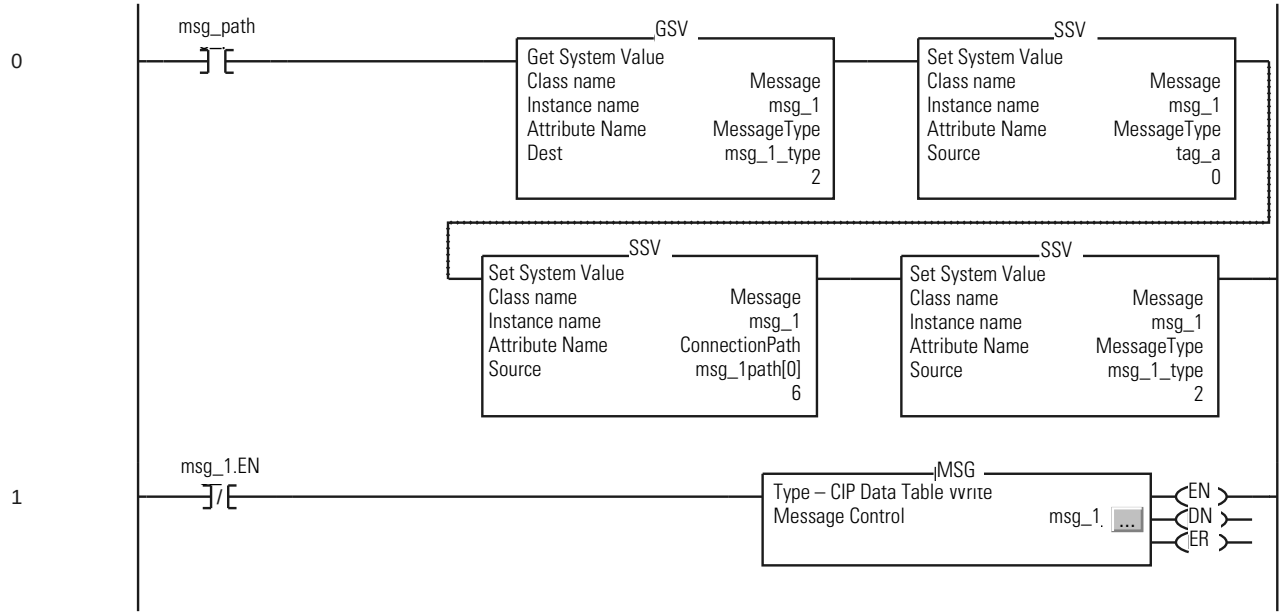
Per modificare un attributo MESSAGE, procedere in questo modo:

1. Utilizzare un'istruzione GSV per ottenere l'attributo MessageType e salvarlo in un tag.
2. Utilizzare un'istruzione SSV per impostare il MessageType su 0.
3. Utilizzare un'istruzione SSV per impostare l'attributo MESSAGE che si desidera modificare.
4. Utilizzare un'istruzione SSV per impostare l'attributo MessageType sul valore originario ottenuto al punto 1.

Esempio: Il seguente esempio modifica l'attributo ConnectionPath, in modo tale che il messaggio venga trasmesso a un controllore diverso. Quando *msg_path* è attivo, imposta il percorso del messaggio *msg _1* al valore di *msg_1_ path*. Questo invia il messaggio a un controllore diverso.

Dove	È
msg_1	il messaggio con l'attributo che si vuole cambiare
msg_1_type	tag che memorizza il valore dell'attributo MessageType
tag_a	tag che memorizza uno 0.
msg_1_path	tag che memorizza il nuovo percorso di collegamento per il messaggio

Ladder



Testo strutturato

```
IF msg_path THEN
     GSV(MESSAGE,msg_1,MessageType,msg_1_type);
     SSV(MESSAGE,msg_1,MessageType,tag_a);
     SSV(MESSAGE,msg_1,ConnectionPath,msg_1_path[0]);
     SSV(MESSAGE,msg_1,MessageType,msg_1_type);
END_IF;
IF NOT msg_1.EN THEN
     MSG(msg_1);
END_IF;
```

Accesso all'oggetto MODULE

L'oggetto MODULE fornisce informazioni sullo stato di un modulo. Per selezionare un determinato oggetto MODULE, impostare nell'operando Nome Oggetto dell'istruzione GSV/SSV il nome del modulo. Il modulo specificato deve essere presente nella sezione Configurazione I/O dell'organizer del controllore e deve avere un nome di dispositivo.

Attributo	Tipo di dati	Istruzione	Descrizione
EntryStatus	INT	GSV	Indica lo stato corrente dell'elemento di mappa specificato. Quando si esegue un'operazione di confronto, i 12 bit più bassi devono essere mascherati. Solo i bit 12–15 sono validi. **Valore:** **Significato:** 16#0000 **Standby:** il controllore è in fase di accensione. 16#1000 **In errore:** errore di una delle connessioni dell'oggetto MODULE associata al modulo. Questo valore non deve essere usato per stabilire se il modulo è in errore poiché l'oggetto MODULE esce da questo stato periodicamente quando tenta di riconnettersi al modulo. Si consiglia invece di verificare lo stato di Esecuzione (16#4000). Controllare che FaultCode sia diverso da 0 per determinare se un modulo è guasto. Se vi è un errore, gli attributi FaultCode e FaultInfo rimangono validi fino a quando la condizione di errore non viene corretta. 16#2000 **Validazione:** l'oggetto MODULE sta verificando l'integrità dell'oggetto MODULE prima di stabilire le connessioni con il modulo. 16#3000 **Connessione in corso:** l'oggetto MODULE sta iniziando le connessioni con il modulo. 16#4000 **In esecuzione:** tutte le connessioni con il modulo sono attive ed i dati vengono trasferiti correttamente. 16#5000 **Chiusura in corso:** l'oggetto MODULE sta eseguendo la chiusura di tutte le connessioni con il modulo. 16#6000 **Inibito:** l'oggetto MODULE è inibito (il bit di inibizione nell'attributo Modalità è impostato). 16#7000 **In attesa:** l'oggetto MODULE principale da cui dipende questo oggetto MODULE non è in esecuzione.
FaultCode	INT	GSV	Un numero che, in caso di errore del modulo, lo identifica.
FaultInfo	DINT	GSV	Fornisce informazioni specifiche sul codice di errore dell'oggetto MODULE.
ForceStatus	INT	GSV	Indica lo stato delle forzature. **Bit:** **Significato:** 0 forzature installate (1=sì, 0-no) 1 forzature abilitate (1=sì, 0=no) 2–15 non usati

Attributo	Tipo di dati	Istruzione	Descrizione
Instance	DINT	GSV	Fornisce il numero di istanza di questo oggetto MODULE.
LEDStatus	INT	GSV	Indica lo stato corrente del LED I/O posto sul frontale del Controllore. **Valore:** **Significato:** 0 **LED spento:** Nessun oggetto MODULE configurato per il Controllore (nella Configurazione I/O dell'organizer del Controllore non è presente alcun modulo). 1 Rosso lampeggiante: nessun oggetto MODULE in esecuzione. 2 Verde lampeggiante: almeno un oggetto MODULE non è in esecuzione. 3 Verde fisso: tutti gli oggetti MODULE sono in esecuzione. **Nota:** non inserire un nome di oggetto con questo attributo in quanto questo attributo si riferisce a tutto il gruppo di moduli.
Mode	INT	GSV	Indica la modalità corrente dell'oggetto MODULE.
		SSV	**Bit:** **Significato:** 0 Se è impostato, provoca un errore grave qualora una delle connessioni dell'oggetto MODULE va in errore mentre il Controllore è in modalità esecuzione (RUN). 2 Se è impostato, fa sì che l'oggetto MODULE passi allo stato Inibito dopo la chiusura di tutte le connessioni con il modulo.

Accesso all'oggetto MOTIONGROUP

L'oggetto MOTIONGROUP fornisce informazioni sullo stato di un gruppo di assi del modulo servo. Specificare il nome del tag del gruppo assi per determinare l'oggetto MOTIONGROUP desiderato.

Attributo	Tipo di dati	Istruzione	Descrizione
Instance	DINT	GSV	Fornisce il numero di istanza di questo oggetto MOTION_GROUP.

Accesso all'oggetto PROGRAM

L'oggetto PROGRAM fornisce informazioni sullo stato di un programma. Specificare il nome del programma per determinare l'oggetto PROGRAM desiderato.

Attributo	Tipo di dati	Istruzione	Descrizione
DisableFlag	SINT	GSV	Controlla l'esecuzione di questo programma.
		SSV	**Valore:** **Significato:** 0 esecuzione abilitata 1 esecuzione disabilitata
Instance	DINT	GSV	Fornisce il numero di istanza di questo oggetto PROGRAM.
LastScanTime	DINT	GSV	Il tempo impiegato per eseguire questo programma l'ultima volta che è stato eseguito. Tempo espresso in microsecondi.
		SSV	
MajorFaultRecord	DINT[11]	GSV	Registra gli errori gravi di questo programma
		SSV	Si consiglia di creare una struttura definita dall'utente per semplificare l'accesso all'attributo MajorFaultRecord:
Nome:	**Tipo dati:**	**Stile:**	**Descrizione:**
TimeLow	DINT	Decimale	i 32 bit più bassi del valore di data e ora dell'errore
TimeHigh	DINT	Decimale	i 32 bit più alti del valore di data e ora dell'errore
Type	INT	Decimale	tipo di errore (programma, I/O, ecc.)
Code	INT	Decimale	codice univoco per l'errore (dipende dal tipo di errore)
Info	DINT[8]	Esadecimale	informazioni specifiche sull'errore (in base al tipo e al codice errore)
MaxScanTime	DINT	GSV	Tempo di esecuzione massimo registrato di questo programma. Tempo espresso in microsecondi.
		SSV	
MinorFaultRecord	DINT[11]	GSV	Registra gli errori minori di questo programma
		SSV	Si consiglia di creare una struttura definita dall'utente per semplificare l'accesso all'attributo MinorFaultRecord:
Nome:	**Tipo dati:**	**Stile:**	**Descrizione:**
TimeLow	DINT	Decimale	i 32 bit più bassi del valore di data e ora dell'errore
TimeHigh	DINT	Decimale	i 32 bit più alti del valore di data e ora dell'errore
Type	INT	Decimale	tipo di errore (programma, I/O, ecc.)
Code	INT	Decimale	codice univoco per l'errore (dipende dal tipo di errore)
Info	DINT[8]	Esadecimale	informazioni specifiche sull'errore (in base al tipo e al codice errore)
SFCRestart	INT	GSV	non usato – riservato per uso futuro
		SSV	

Accesso all'oggetto Routine

L'oggetto ROUTINE fornisce informazioni sullo stato di una routine. Specificare il nome della routine per determinare l'oggetto ROUTINE desiderato.

Attributo	Tipo di dati	Istruzione	Descrizione
Instance	DINT	GSV	Fornisce il numero di istanza di questo oggetto ROUTINE. Valori validi sono 0–65 535.

Accesso all'oggetto SERIALPORT

L'oggetto SERIALPORT fornisce un'interfaccia alla porta di comunicazione seriale.

Attributo	Tipo di dati	Istruzione	Descrizione
BaudRate	DINT	GSV	Specifica la velocità di trasmissione (baud rate). Valori validi sono 110, 300, 600, 1200, 2400, 4800, 9600 e 19200 (default).
DataBits	SINT	GSV	Indica il numero di bit di dati per carattere. **Valore:** **Significato:** 7 7 bit dati (solo ASCII) 8 8 bit dati (default)
Parity	SINT	GSV	Specifica la parità. **Valore:** **Significato:** 0 nessuna parità (no default) 1 parità dispari (solo ASCII) 2 parità pari
RTSOffDelay	INT	GSV	Il tempo di ritardo per la disattivazione della linea RTS (Request To Send) dopo che l'ultimo carattere è stato trasmesso. valore valido 0–32 767. Ritardo in conteggi di 20 msec. Il valore di default è 0 msec.
RTSSendDelay	INT	GSV	Il tempo di ritardo per la trasmissione del primo carattere di un messaggio dopo l'attivazione della linea RTS. valore valido 0–32 767. Ritardo in conteggi di 20 msec. Il valore di default è 0 msec.
StopBits	SINT	GSV	Indica il numero di bit di stop. **Valore:** **Significato:** 1 1 bit di stop (default) 2 2 bit stop (solo ASCII)
PendingBaudRate	DINT	SSV	Valore in attesa dell'attributo BaudRate.
PendingDataBits	SINT	SSV	Valore in attesa dell'attributo DataBits.
PendingParity	SINT	SSV	Valore in attesa dell'attributo Parity.

Attributo	Tipo di dati	Istruzione	Descrizione
PendingRTSOffDelay	INT	SSV	Valore in attesa dell'attributo RTSOffDelay.
PendingRTSSendDelay	INT	SSV	Valore in attesa dell'attributo RTSSendDelay.
PendingStopBits	SINT	SSV	Valore in attesa dell'attributo StopBits.

Per applicare i valori per uno degli attributi di SERIALPORT in attesa:

1. Utilizzare un'istruzione SSV per impostare il valore per l'attributo in attesa.

   È possibile impostare tutti gli attributi in attesa che si desidera utilizzando un'istruzione SSV per ogni attributo in attesa.

2. Utilizzare un'istruzione MSG per applicare il valore. L'istruzione MSG si applica ad ogni attributo in attesa che si è impostato. Configurare le istruzioni MSG come segue:

Scheda di configurazione MSG	Campo	Valore
Configuration	Message Type	CIP generico
	Service Code	0d esad.
	Object Type	6f esad.
	Object ID	1
	Object attribute	lasciare vuoto
	Source	lasciare vuoto
	Number of Elements	0
	Destination	lasciare vuoto
Comunication	Path	percorso di comunicazione verso se stesso (1,*s* dove *s* = numero di slot del Controllore)

Accesso all'oggetto TASK

L'oggetto TASK fornisce informazioni sullo stato di un task. Specificare il nome del task per determinare l'oggetto TASK desiderato.

Attributo	Tipo di dati	Istruzione	Descrizione
DisableUpdateOutputs	DINT	GSV SSV	Abilita o disabilita l'elaborazione delle uscite al termine del task **Per:** / **Impostare l'attributo su:** abilita l'elaborazione delle uscite al termine del task: 0 disabilita l'elaborazione delle uscite al termine del task: 1 (o qualsiasi valore diverso da zero)
EnableTimeOut	DINT	GSV SSV	Abilita o disabilita la funzione di timeout di un task evento. **Per:** / **Impostare l'attributo su:** disabilitare la funzione di timeout: 0 abilitare la funzione di timeout: 1 (o qualsiasi valore diverso da zero)
InhibitTask	DINT	GSV SSV	Impedisce l'esecuzione del task. Se un task è inibito, il controllo esegue la prescansione del task quando il controllore passa dalla modalità di programmazione a quella di esecuzione o test. **Per:** / **Impostare l'attributo su:** abilitare il task: 0 (default) inibire (disabilitare) il task: 1 (o qualsiasi valore diverso da zero)
Instance	DINT	GSV	Fornisce il numero di istanza di questo oggetto TASK. Valori validi sono 0–31.
LastScanTime	DINT	GSV SSV	Il tempo impiegato per eseguire questo task l'ultima volta che è stato eseguito. Tempo espresso in microsecondi.
MaxInterval	DINT[2]	GSV SSV	L'intervallo di tempo massimo tra esecuzioni successive del task. DINT[0] contiene i 32 bit meno significativi del valore; DINT[1] contiene i 32 bit più significativi del valore. Un valore di 0 indica 1 o meno esecuzioni del task.
MaxScanTime	DINT	GSV SSV	Tempo di esecuzione massimo registrato di questo programma. Tempo espresso in microsecondi.
MinInterval	DINT[2]	GSV SSV	L'intervallo di tempo minimo tra esecuzioni successive del task. DINT[0] contiene i 32 bit meno significativi del valore; DINT[1] contiene i 32 bit più significativi del valore. Un valore di 0 indica 1 o meno esecuzioni del task.
OverlapCount	DINT	GSV SSV	Numero di volte in cui il task è stato attivato mentre era ancora in esecuzione. Valido per un evento o un task periodico. Per azzerare il conteggio, impostare l'attributo su 0.
Priority	INT	GSV SSV	La priorità relativa di questo task in confronto agli altri task. Valori validi sono 1...15.

Attributo	Tipo di dati	Istruzione	Descrizione	
Rate	DINT	GSV SSV	**Se il tipo di task è:**	**L'attributo Rate specifica:**
			periodico	Il periodo del task. Tempo espresso in microsecondi.
			evento	Il valore di timeout del task. Tempo espresso in microsecondi.
StartTime	DINT[2]	GSV SSV	Il valore del WALLCLOCKTIME quando l'ultima esecuzione del task ha avuto inizio. DINT[0] contiene i 32 bit meno significativi del valore; DINT[1] contiene i 32 bit più significativi del valore.	
Status	DINT	GSV SSV	Fornisce informazioni sullo stato del task. Una volta che il controllore imposta uno di questi bit, il bit deve essere azzerato manualmente.	
			Per determinare se:	**Esaminare questo bit:**
			Un'istruzione EVNT ha attivato il task (solo task evento).	0
			Un timeout ha attivato il task (solo task evento).	1
			Si è verificata una sovrapposizione per questo task.	2
Watchdog	DINT	GSV SSV	Tempo limite per l'esecuzione di tutti i programmi associati a questo task. Tempo espresso in microsecondi. Se si inserisce 0, vengono assegnati questi valori:	
			Tempo:	**Tipo di task:**
			0,5 sec	periodico o evento
			5,0 sec	continuo

Accesso all'oggetto WALLCLOCKTIME

L'oggetto WALLCLOCKTIME fornisce una registrazione cronologica che il controllore utilizza per la schedulazione.

Attributo	Tipo di dati	Istruzione	Descrizione
CSTOffset	DINT[2]	GSV SSV	Offset positivo dal CurrentValue dell'oggetto CST (tempo di sistema coordinato, vedere pagina 179). DINT[0] contiene i 32 bit meno significativi del valore; DINT[1] contiene i 32 bit più significativi del valore. Valore in μsecs. Il valore di default è 0.
CurrentValue	DINT[2]	GSV SSV	Il valore corrente dell'ora dell'orologio. DINT[0] contiene i 32 bit meno significativi del valore; DINT[1] contiene i 32 bit più significativi del valore. Il valore rappresenta il numero di microsecondi trascorsi dalle 0000 del 1 Gennaio 1972. Gli oggetti CST e WALLCLOCKTIME sono matematicamente correlati nel Controllore. Ad esempio, se si sommano il CurrentValue del CST e il CTSOffset del WALLCLOCKTIME, il risultato è il CurrentValue del WALLCLOCKTIME.
DateTime	DINT[7]	GSV SSV	La data e l'ora in formato leggibile. DINT[0] anno DINT[1] intero rappresentante il mese (1–12) DINT[2] intero rappresentante il giorno (1–31) DINT[3] ora (0–23) DINT[4] minuto (0–59) DINT[5] secondi (0–59) DINT[6] microsecondi (0–999 999)

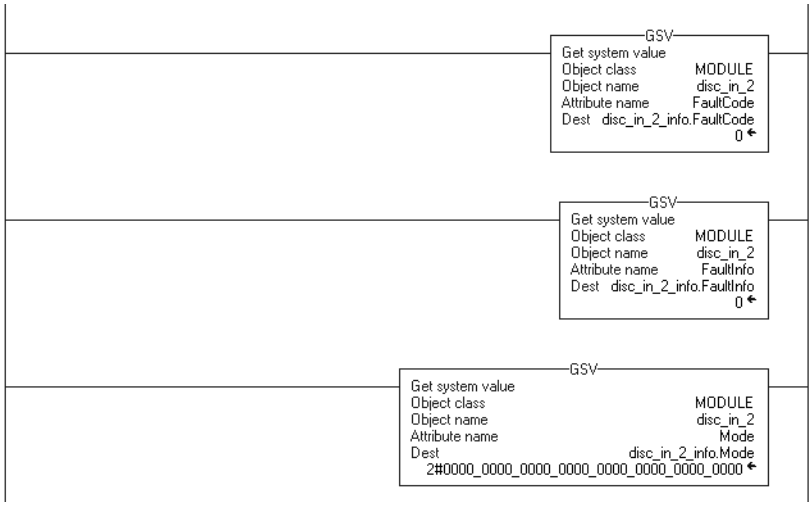
Esempio di programmazione GSV/SSV

Get Fault Information

Nei seguenti esempi si utilizzano le istruzioni GSV per ottenere informazioni sugli errori.

Esempio 1: In questo esempio si mostra come ottenere informazioni sugli errori dal *disc_in_2* del modulo I/O ed inserire i dati in una struttura definita dall'utente *disc_in_2_info*.

Ladder

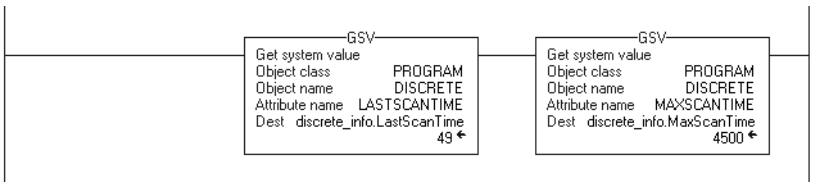


Testo strutturato

```
GSV(MODULE,disc_in_2,FaultCode,disc_in_2_info.FaultCode);

GSV(MODULE,disc_in_2,FaultInfo,disc_in_2_info.FaultInfo);

GSV(MODULE,disc_in_2,Mode,disc_in_2info.Mode);
```

Esempio 2: In questo esempio si mostra come ottenere informazioni sullo stato di un programma *discrete* ed inserire i dati in una struttura definita dall'utente *discrete_info*.

Ladder

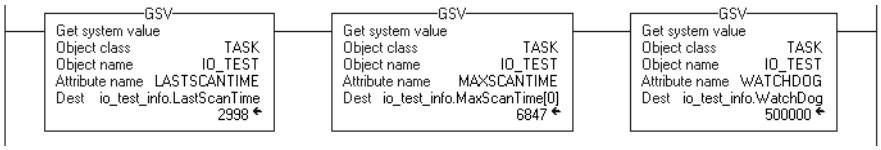


Testo strutturato

```
GSV(PROGRAM,DISCRETE,LASTSCANTIME,
     discrete_info.LastScanTime);

GSV(PROGRAM,DISCRETE,MAXSCANTIME,discrete_info.MaxScanTime);
```

Esempio 3: In questo esempio si mostra come ottenere informazioni sullo stato di un task *IO_test* ed inserire i dati in una struttura definita dall'utente *io_test_info*.

Ladder



Testo strutturato

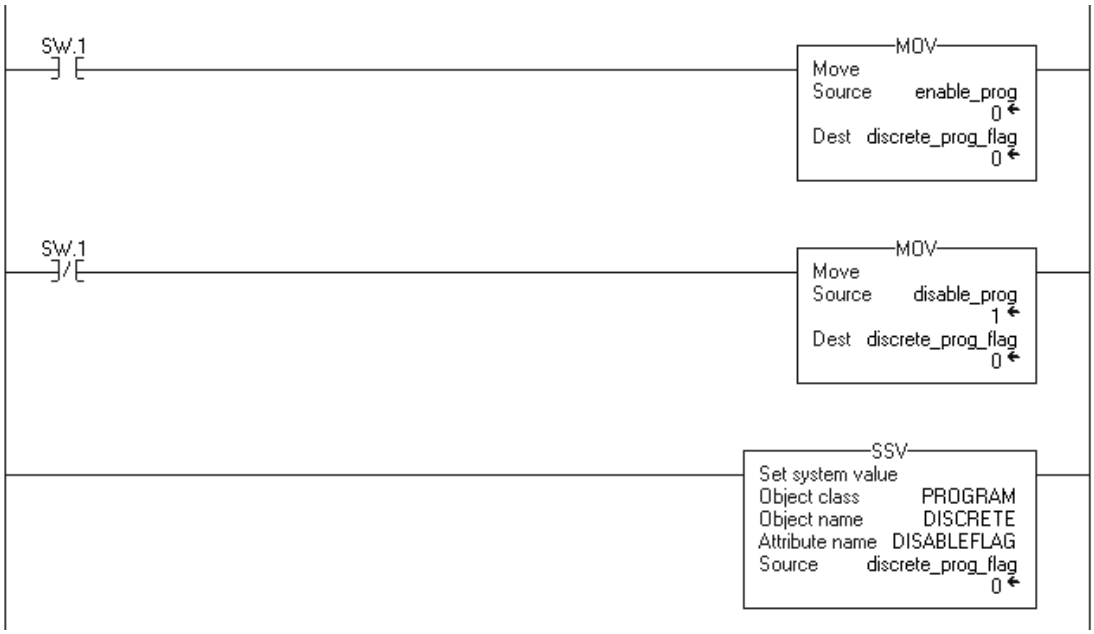
```
GSV(TASK,IO_TEST,LASTSCANTIME,io_test_info.LastScanTime);

GSV(TASK,IO_TEST,MAXSCANTIME,io_test_info.MaxScanTime);

GSV(TASK,IO_TEST,WATCHDOG,io_test_info.WatchDog);
```

Impostazione degli indicatori di abilitazione e disabilitazione

Nel seguente esempio si utilizza l'istruzione SSV per abilitare o disabilitare un programma. È possibile utilizzare questo metodo anche per abilitare o disabilitare un modulo I/O; una soluzione di programma simile a quella che prevede l'uso dei bit di inibizione con un processore PLC-5.

Esempio: In base allo stato di *SW.1*, inserire il valore appropriato nell'attributo *disableflag* del programma *discrete*.

Ladder



Testo strutturato

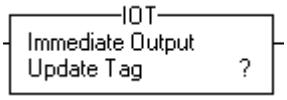
```
IF SW.1 THEN

     discrete_prog_flag := enable_prog;

ELSE

     discrete_prog_flag := disable_prog;

END_IF;

SSV(PROGRAM,DISCRETE,DISABLEFLAG,discrete_prog_flag);
```

Immediate Output (IOT)

L'istruzione IOT aggiorna immediatamente i dati di uscita specificati (tag di uscita o tag prodotto).

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Update Tag		tag	tag che si desidera aggiornare: • tag di uscita di un modulo I/O oppure • tag prodotto *Non* scegliere un membro o un elemento di un tag. Ad esempio, Local:5:0 va bene, ma Local:5:0.Data *non* è accettabile.

Testo strutturato

```
IOT(tag_uscita);
```

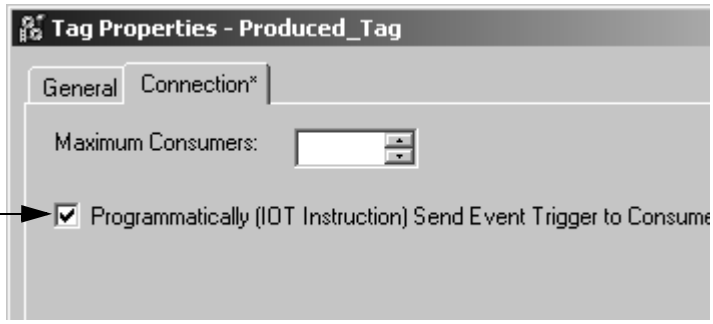
Gli operandi sono uguali a quelli utilizzati per l'istruzione IOT ladder.

Descrizione:

L'istruzione IOT ha la precedenza sull'intervallo di pacchetto richiesto (RPI) di una connessione di uscita e invia dati nuovi attraverso la connessione.

- Una connessione di uscita è una connessione associata al tag di uscita di un modulo I/O o a un tag prodotto.
- Se la connessione è per un tag prodotto, l'istruzione IOT invia anche il trigger evento al controllore consumatore. Questo consente all'istruzione IOT di attivare un task evento nel controllore consumatore.

Per usare un'istruzione IOT e il tag prodotto per attivare un task evento in un controllore consumatore, configurare il tag prodotto nel modo seguente:



Spuntare questa casella.

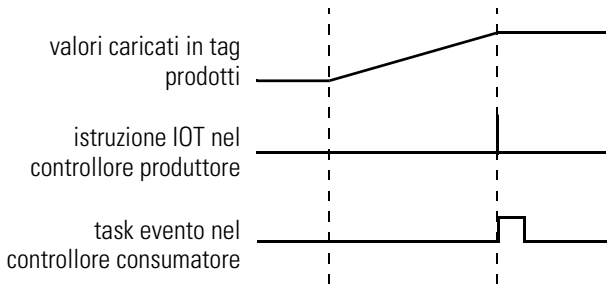
In questo modo il tag viene configurato per aggiornare il trigger evento solo tramite istruzione IOT.

Il tipo di rete tra controllori determina quando il controllore consumatore riceve i nuovi dati e il trigger evento tramite l'istruzione IOT.

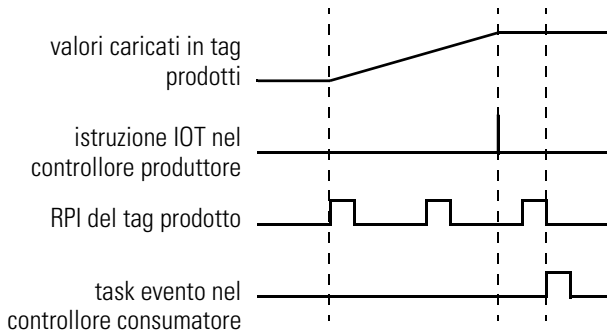
Con questo controllore	Tramite questa rete	Il dispositivo consumatore riceve i dati e il trigger evento
ControlLogix	backplane	immediatamente
	rete EtherNet/IP	immediatamente
	rete ControlNet	entro l'intervallo pacchetto effettivo (API) del tag consumato (connessione)
SoftLogix5800	È possibile produrre e consumare tag solo tramite una rete ControlNet.	entro l'intervallo pacchetto effettivo (API) del tag consumato (connessione)

Lo schema seguente confronta la ricezione di dati tramite un'istruzione IOT attraverso le reti EtherNet/IP e ControlNet.

rete EtherNet/IP



rete ControlNet


valori caricati in tag prodotti
istruzione IOT nel controllore produttore
RPI del tag prodotto
task evento nel controllore consumatore

Indicatori di stato aritmetico: non influenzati

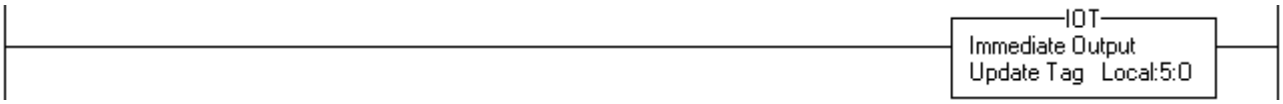
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn è impostato	na	EnableIn è sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione: • aggiorna la connessione del tag specificato • azzera il timer RPI della connessione	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio 1: Quando viene eseguita l'istruzione IOT, invia immediatamente i valori del tag *Local:5:0* al modulo di uscita.

Ladder

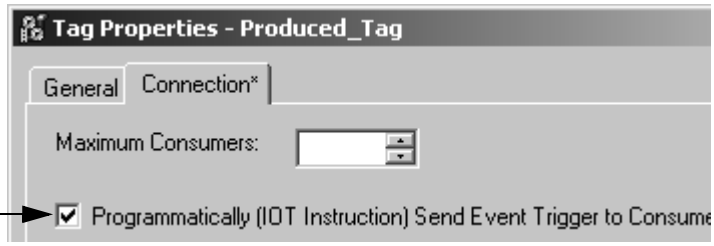


Testo strutturato

```
IOT (Local:5:O);
```

Esempio 2: Il controllore controlla la stazione 24 e produce dati per la stazione successiva (stazione 25). Per usare un'istruzione IOT per segnalare la trasmissione di nuovi dati, il tag prodotto viene configurato nel modo seguente:

Produced_Tag è configurato per aggiornare il trigger evento tramite un'istruzione IOT.

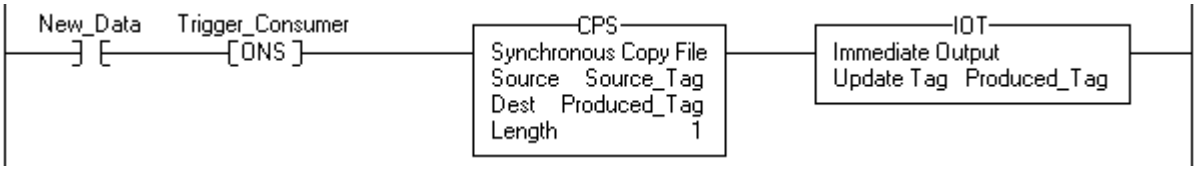


Ladder

Se *New_Data* = on, avviene quando segue per una scansione:

L'istruzione CPS imposta *Produced_Tag* = *Source_Tag*.

L'istruzione IOT aggiorna *Produced_Tag* e invia tale aggiornamento al controllore consumatore (stazione 25). Quando il controllore consumatore riceve l'aggiornamento, aziona il task evento associato in quel controllore.



Testo strutturato

```
IF New_Data AND NOT Trigger_Consumer THEN
  CPS (Source_Tag,Produced_Tag,1);
  IOT (Produced_Tag);
END_IF;
Trigger_Consumer := New_Data;
```

Istruzioni di confronto

(CMP, EQU, GEQ, GRT, LEQ, LES, LIM, MEQ, NEQ)

Introduzione

Le istruzioni di confronto permettono di confrontare dei valori utilizzando un'espressione o un'istruzione di confronto specifica.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
confrontare dei valori con un'espressione	CMP	ladder testo strutturato(1)	204
verificare se due valori sono uguali	EQU	ladder testo strutturato(2) blocco funzione	209
verificare se un valore è maggiore o uguale ad un secondo valore	GEQ	ladder testo strutturato(1) blocco funzione	213
verificare se un valore è maggiore di un secondo valore	GRT	ladder testo strutturato(1) blocco funzione	217
verificare se un valore è minore o uguale ad un secondo valore	LEQ	ladder testo strutturato(1) blocco funzione	221
verificare se un valore è minore di un secondo valore	LES	ladder testo strutturato(1) blocco funzione	225
verificare se un valore è compreso tra due altri valori	LIM	ladder testo strutturato(1) blocco funzione	229
filtrare due valori attraverso una mask e vedere se sono uguali	MEQ	ladder testo strutturato(1) blocco funzione	235
verificare se un valore è diverso da un secondo valore	NEQ	ladder testo strutturato(1) blocco funzione	240

(1) Non esiste un'istruzione equivalente in testo strutturato. Utilizzare un'altra programmazione in testo strutturato per ottenere lo stesso risultato. Vedere la descrizione dell'istruzione.

(2) Non esiste un'istruzione equivalente in testo strutturato. Utilizzare l'operatore in un'espressione.

È possibile confrontare valori di tipo diverso, ad esempio a virgola mobile ed interi.

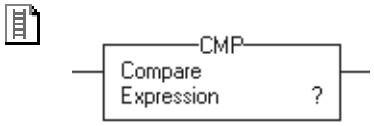
Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimali, in genere DINT o REAL.

Compare (CMP)

L'istruzione CMP esegue un confronto delle operazioni aritmetiche specificate nell'espressione.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Espressione	SINT	immediato	un'espressione formata da tag e/o valori immediati separati da operatori
	INT	tag	
	DINT		
	REAL		
	stringa		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		

Testo strutturato

Il testo strutturato non ha un'istruzione CMP, ma è possibile ottenere gli stessi risultati usando un costrutto IF...THEN e un'espressione.

```
IF espressione_booleana THEN
    <istruzione>;
END_IF;
```

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi di costrutti ed espressioni nel testo strutturato.

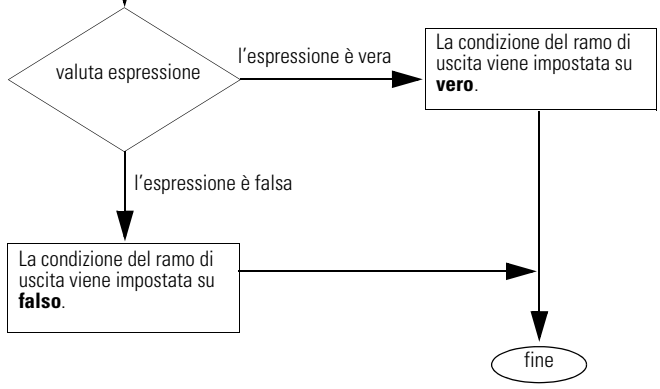
Descrizione: Definire l'espressione CMP utilizzando operatori, tag e valori immediati. Utilizzare delle parentesi ( ) per definire parti di espressioni più complesse.

L'esecuzione di un'istruzione CMP è leggermente più lenta ed utilizza più memoria rispetto all'esecuzione di altre istruzioni di confronto. Il vantaggio di un'istruzione CMP consiste nel fatto che consente di inserire espressioni complesse in una istruzione.

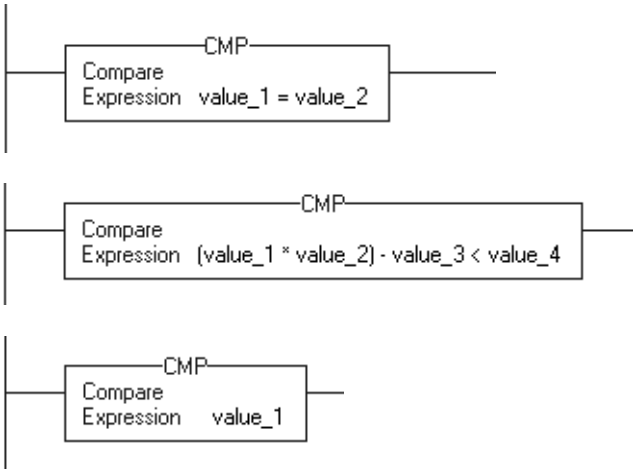
Indicatori di stato aritmetico: L'istruzione CMP attiva gli indicatori di stato aritmetico soltanto se l'espressione contiene un operatore (ad esempio, +, –, *, /) che attiva gli indicatori di stato aritmetico.

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempi: Se l'istruzione CMP trova che l'espressione è vera, la condizione del ramo di uscita viene impostata su vero.



Se si inserisce un'espressione senza un operatore di confronto, ad esempio *valore_1 + valore_2* oppure *valore_1*, l'istruzione valuta l'espressione nel modo seguente:

Se l'espressione è	La condizione del ramo di uscita è impostata su
diversa zero	vero
zero	falso

Espressioni CMP

Le espressioni delle istruzioni CMP vanno programmate nello stesso modo delle espressioni delle istruzioni FSC. Per informazioni su operatori validi, formato ed ordine delle operazioni comuni ad entrambe le istruzioni, utilizzare le seguenti sezioni.

Operatori validi

Operatore	Descrizione	Ottimale
+	somma	DINT, REAL
–	sottrai/nega	DINT, REAL
*	moltiplica	DINT, REAL
/	dividi	DINT, REAL
=	uguale	DINT, REAL
<	minore di	DINT, REAL
<=	minore di o uguale	DINT, REAL
>	maggiore di	DINT, REAL
>=	maggiore di o uguale a	DINT, REAL
<>	diverso	DINT, REAL
**	esponente (x alla y)	DINT, REAL
ABS	valore assoluto	DINT, REAL
ACS	arcocoseno	REAL
AND	AND di bit	DINT
ASN	arcoseno	REAL
ATN	arcotangente	REAL
COS	coseno	REAL

Operatore	Descrizione	Ottimale
DEG	da radianti a gradi	DINT, REAL
FRD	da BCD a intero	DINT
LN	logaritmo naturale	REAL
LOG	logaritmo in base 10	REAL
MOD	modulo-divide	DINT, REAL
NOT	complemento di bit	DINT
OR	OR di bit	DINT
RAD	da gradi a radianti	DINT, REAL
SIN	seno	REAL
SQR	radice quadrata	DINT, REAL
TAN	tangente	REAL
TOD	da intero a BCD	DINT
TRN	tronca	DINT, REAL
XOR	OR di bit esclusivo	DINT

Formattazione delle espressioni

Per ogni operatore utilizzato in un'espressione, è necessario fornire uno o due operandi (tag o valori immediati). Utilizzare la seguente tabella per formare operatori ed operandi in un'espressione:

Per operatori con	Utilizzare questo Formato	Esempi
un operando	operatore(operando)	ABS(*tag_a*)
due operandi	operando_a operatore operando_b	• *tag_b* + 5 • *tag_c* AND *tag_d* • (*tag_e* ** 2) MOD (*tag_f*/*tag_g*)

Determinazione dell'ordine delle operazioni

Le operazioni che vengono scritte nell'espressione sono eseguite dall'istruzione in un determinato ordine, non necessariamente nell'ordine con cui sono state scritte. È possibile cambiare l'ordine delle operazioni raggruppando i termini all'interno di parentesi, forzando l'istruzione a eseguire un'operazione all'interno delle parentesi prima di altre operazioni.

Operazioni di ordine uguale vengono eseguite da sinistra a destra.

Ordine	Funzionamento
1.	( )
2.	ABS, ACS, ASN, ATN, COS, DEG, FRD, LN, LOG, RAD, SIN, SQR, TAN, TOD, TRN
3.	**
4.	– (nega), NOT
5.	*, /, MOD
6.	<, <=, >, >=, =
7.	– (sottrai), +
8.	AND
9.	XOR
10.	OR

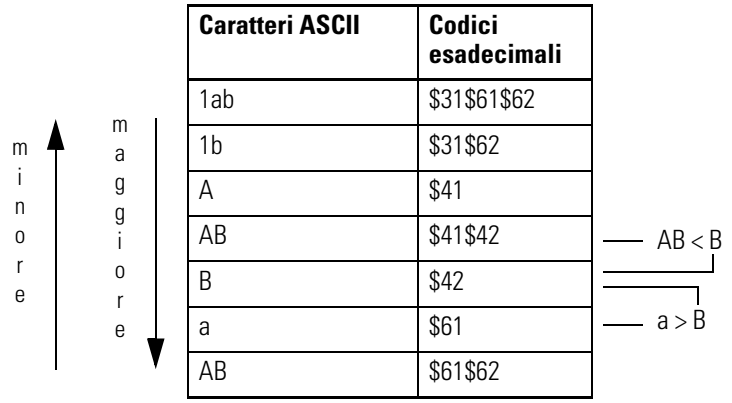
Uso di stringhe in un'espressione

Utilizzare un'espressione ladder o di testo strutturato per confrontare i tipi di dati delle stringhe. Per utilizzare stringhe in un'espressione, attenersi alle seguenti regole:

- Un'espressione consente di confrontare due tag stringa.
- *Non* è possibile inserire caratteri ASCII direttamente nell'espressione.
- Sono permessi solamente i seguenti operatori

Operatore	Descrizione
=	uguale
<	minore di
<=	minore di o uguale a
>	maggiore di
>=	maggiore di o uguale a
<>	diverso

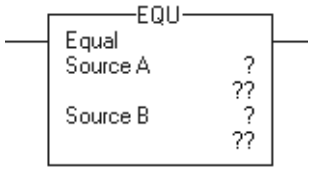
- Le stringhe sono uguali se i loro caratteri corrispondono.
- I caratteri ASCII sono sensibili alle maiuscole. "A" maiuscola ($41) *non* è uguale ad "a" minuscola ($61).
- I valori esadecimali dei caratteri determinano se una stringa è minore o maggiore di un'altra stringa. Per il codice esadecimale di un carattere, vedere il retro copertina di questo manuale.
- Quando le due stringhe sono ordinate come in un elenco telefonico, l'ordine delle stringhe determina quale è la maggiore.



Equal to (EQU)

L'istruzione EQU verifica se Source A è uguale a Source B.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore da confrontare con Source B
	INT	tag	
	DINT		
	REAL		
	stringa		
Source B	SINT	immediato	valore da confrontare con Source A
	INT	tag	
	DINT		
	REAL		
	stringa		

- Se si inserisce un tag SINT oppure INT, il valore viene convertito in un valore DINT con un'estensione di segno.
- I valori REAL raramente sono completamente uguali. Se si desidera determinare l'uguaglianza di due valori REAL, utilizzare l'istruzione LIM.
- I tipi di dati stringa sono:
  - tipo dati STRINGA di default
  - qualunque nuovo tipo di dati stringa che si crea
- Per verificare i caratteri di una stringa, immettere un tag stringa per Source A e Source B.

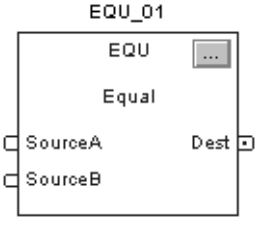
Testo strutturato

```
IF sourceA = sourceB THEN
    <istruzioni>;
```

Usare il segno di uguale "=" come operatore all'interno di un'espressione. Questa espressione valuta se *sourceA* è uguale a *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag EQU	FBD_COMPARE	struttura	struttura EQU

Struttura FBD_ COMPARE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	REAL	Valore da confrontare con SourceB. Valori validi = qualsiasi numero a virgola mobile
SourceB	REAL	Valore da confrontare con SourceA. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	BOOL	Risultato dell'istruzione. Equivale alla condizione del ramo di uscita dell'istruzione ladder EQU.

Descrizione: Usare l'istruzione EQU per confrontare due numeri o due stringhe di caratteri ASCII. Quando si confrontano stringhe:

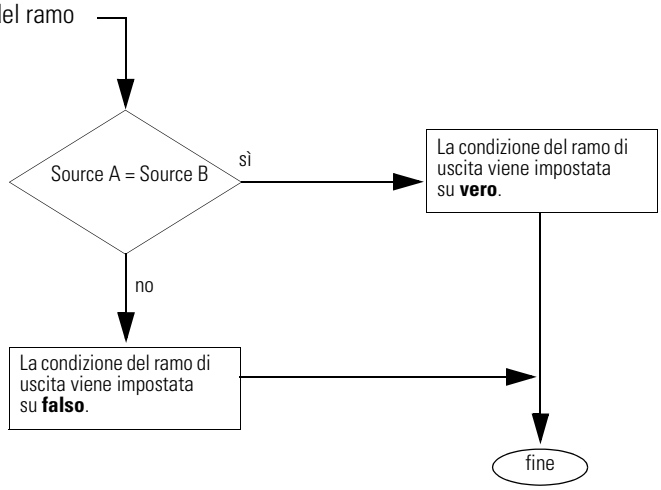
- Le stringhe sono uguali se i loro caratteri corrispondono.
- I caratteri ASCII sono sensibili alle maiuscole. "A" maiuscola ($41) *non* è uguale ad "a" minuscola ($61).

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

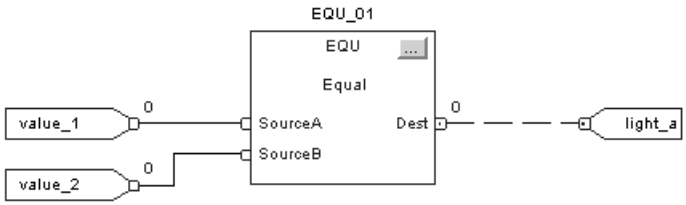
Esempio: Se *valore_1* è uguale a *valore_2*, impostare *light_a*. Se *valore_1* non è uguale a *valore_2*, azzerare *light_a*.

Ladder

Testo strutturato

```
light_a := (valore_1 = valore_2);
```

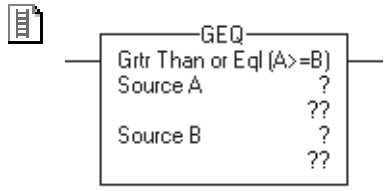
Blocco funzione



Greater than or Equal to (GEQ)

L'istruzione GEQ verifica se Source A è maggiore di o uguale a Source B.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore da confrontare con Source B
	INT	tag	
	DINT		
	REAL		
	stringa		
Source B	SINT	immediato	valore da confrontare con Source A
	INT	tag	
	DINT		
	REAL		
	stringa		

- Se si inserisce un tag SINT oppure INT, il valore viene convertito in un valore DINT con un'estensione di segno.
- I tipi di dati stringa sono:
  - tipo dati STRINGA di default
  - qualunque nuovo tipo di dati stringa che si crea
- Per verificare i caratteri di una stringa, immettere un tag stringa per Source A e Source B.

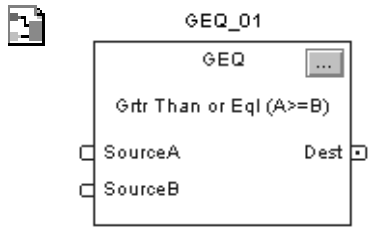
Testo strutturato

```
IF sourceA >= sourceB THEN
    <istruzioni>;
```

Usare segni di maggiore e uguale adiacenti ">=" come operatore all'interno di un'espressione. Questa espressione valuta se *sourceA* è maggiore di o uguale a *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag GEQ	FBD_COMPARE	struttura	struttura GEQ

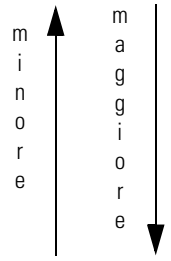
Struttura FBD_ COMPARE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	REAL	Valore da confrontare con SourceB. Valori validi = qualsiasi numero a virgola mobile
SourceB	REAL	Valore da confrontare con SourceA. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	BOOL	Risultato dell'istruzione. Equivale alla condizione del ramo di uscita dell'istruzione ladder GEQ.

Descrizione: L'istruzione GEQ verifica se Source A è maggiore di o uguale a Source B.

Quando si confrontano stringhe:

- I valori esadecimali dei caratteri determinano se una stringa è minore o maggiore di un'altra stringa. Per il codice esadecimale di un carattere, vedere il retro copertina di questo manuale.
- Quando le due stringhe sono ordinate come in un elenco telefonico, l'ordine delle stringhe determina quale è la maggiore.



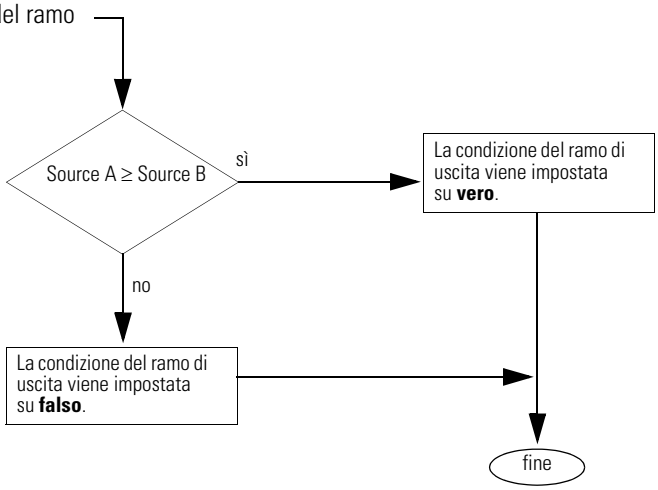
Caratteri ASCII	Codici esadecimali	
1ab	$31$61$62	
1b	$31$62	
A	$41	
AB	$41$42	AB < B
B	$42	
a	$61	a > B
AB	$61$62	

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Ladder

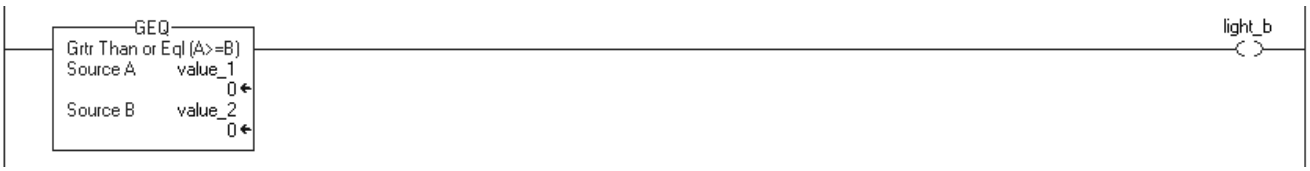
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Se *valore_1* è maggiore di o uguale a *valore_2*, impostare *light_b*. Se *valore_1* è minore di *valore_2*, azzerare *light_b*.

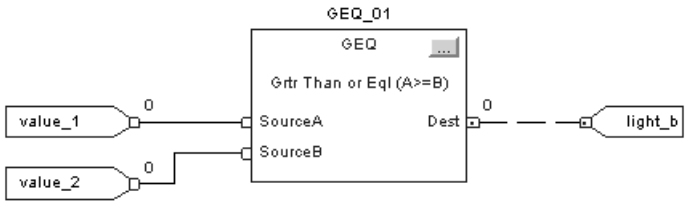
Ladder



Testo strutturato

```
light_b := (valore_1 >= valore_2);
```

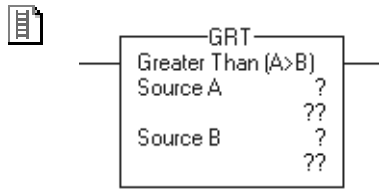
Blocco funzione



Greater Than (GRT)

L'istruzione GRT verifica se Source A è maggiore di Source B.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore da confrontare con Source B
	INT	tag	
	DINT		
	REAL		
	stringa		
Source B	SINT	immediato	valore da confrontare con Source A
	INT	tag	
	DINT		
	REAL		
	stringa		

- Se si inserisce un tag SINT oppure INT, il valore viene convertito in un valore DINT con un'estensione di segno.
- I tipi di dati stringa sono:
  - tipo dati STRINGA di default
  - qualunque nuovo tipo di dati stringa che si crea
- Per verificare i caratteri di una stringa, immettere un tag stringa per Source A e Source B.

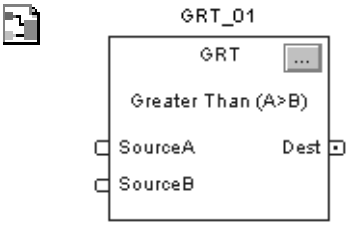
Testo strutturato

```
IF sourceA > sourceB THEN
    <istruzioni>;
```

Usare il segno di maggiore ">" come operatore all'interno dell'espressione. Questa espressione valuta se *sourceA* è maggiore di *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag GRT	FBD_COMPARE	struttura	struttura GRT

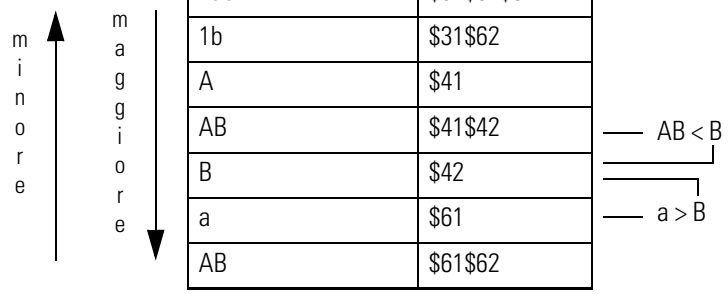
Struttura FBD_ COMPARE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	REAL	Valore da confrontare con SourceB. Valori validi = qualsiasi numero a virgola mobile
SourceB	REAL	Valore da confrontare con SourceA. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	BOOL	Risultato dell'istruzione. Equivale alla condizione del ramo di uscita dell'istruzione ladder GRT.

Descrizione: L'istruzione GRT verifica se Source A è maggiore di Source B.

Quando si confrontano stringhe:

- I valori esadecimali dei caratteri determinano se una stringa è minore o maggiore di un'altra stringa. Per il codice esadecimale di un carattere, vedere il retro copertina di questo manuale.
- Quando le due stringhe sono ordinate come in un elenco telefonico, l'ordine delle stringhe determina quale è la maggiore.



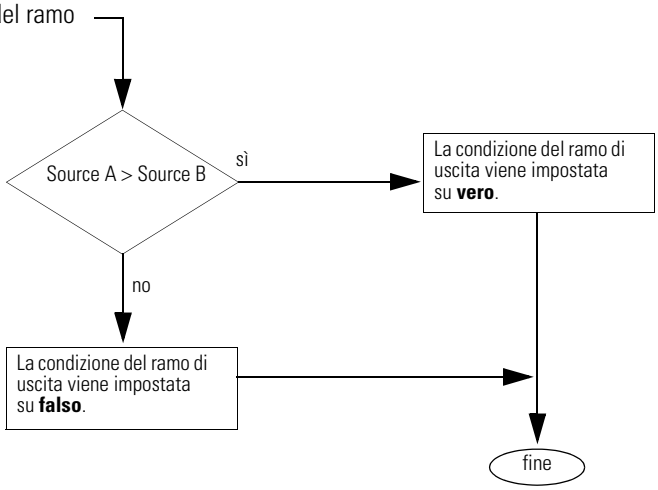
Caratteri ASCII	Codici esadecimali
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
AB	$61$62

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

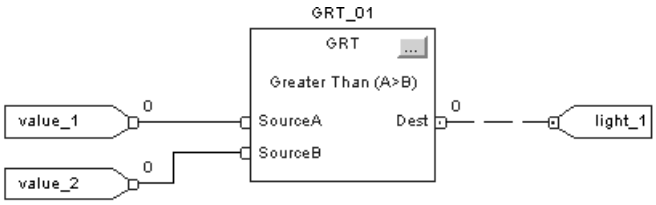
Esempio: Se *valore_1* è maggiore di *valore_2*, impostare *light_1*. Se *valore_1* è minore uguale a *valore_2*, azzerare *light_1*.

Ladder

Testo strutturato

```
light_1 := (valore_1 > valore_2);
```

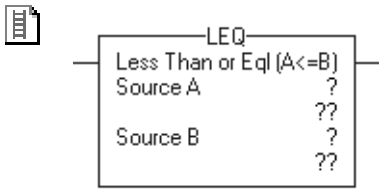
Blocco funzione



Less Than or Equal to (LEQ)

L'istruzione LEQ verifica se Source A è minore di o uguale a Source B.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore da confrontare con Source B
	INT	tag	
	DINT		
	REAL		
	stringa		
Source B	SINT	immediato	valore da confrontare con Source A
	INT	tag	
	DINT		
	REAL		
	stringa		

- Se si inserisce un tag SINT oppure INT, il valore viene convertito in un valore DINT con un'estensione di segno.
- I tipi di dati stringa sono:
  - tipo dati STRINGA di default
  - qualunque nuovo tipo di dati stringa che si crea
- Per verificare i caratteri di una stringa, immettere un tag stringa per Source A e Source B.

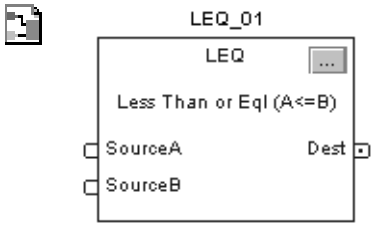
Testo strutturato

```
IF sourceA <= sourceB THEN
    <istruzioni>;
```

Usare segni di minore e uguale adiacenti "<=" come operatore all'interno dell'espressione. Questa espressione valuta se *sourceA* è minore di o uguale a *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag LEQ	FBD_COMPARE	struttura	struttura LEQ

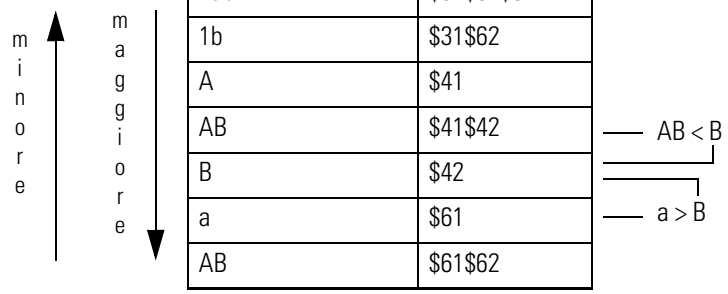
Struttura FBD_ COMPARE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	REAL	Valore da confrontare con SourceB. Valori validi = qualsiasi numero a virgola mobile
SourceB	REAL	Valore da confrontare con SourceA. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	BOOL	Risultato dell'istruzione. Equivale alla condizione del ramo di uscita dell'istruzione ladder LEQ.

Descrizione: L'istruzione LEQ verifica se Source A è minore di o uguale a Source B.

Quando si confrontano stringhe:

- I valori esadecimali dei caratteri determinano se una stringa è minore o maggiore di un'altra stringa. Per il codice esadecimale di un carattere, vedere il retro copertina di questo manuale.
- Quando le due stringhe sono ordinate come in un elenco telefonico, l'ordine delle stringhe determina quale è la maggiore.



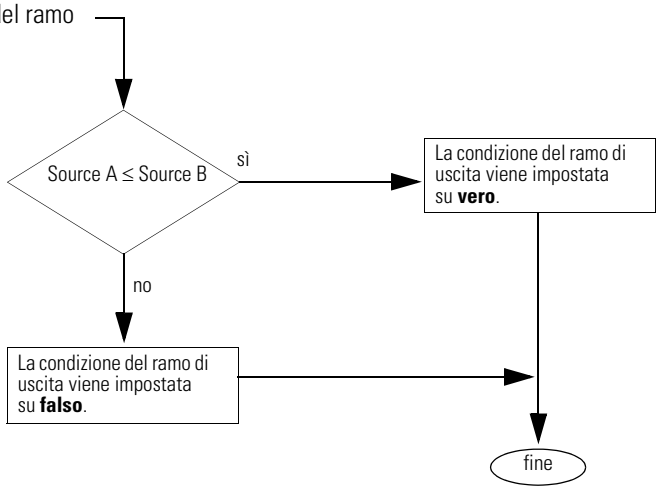
Caratteri ASCII	Codici esadecimali
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
AB	$61$62

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

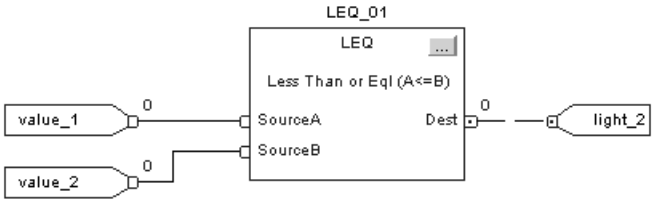
Esempio: Se *valore_1* è minore o uguale a *valore_2*, impostare *light_2*. Se *valore_1* è maggiore di *valore_2*, azzerare *light_2*.

Ladder

Testo strutturato

```
light_2 := (valore_1 <= valore_2);
```

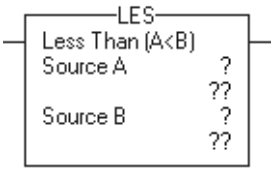
Blocco funzione



Less Than (LES)

L'istruzione LES verifica se Source A è minore di Source B.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore da confrontare con Source B
	INT	tag	
	DINT		
	REAL		
	stringa		
Source B	SINT	immediato	valore da confrontare con Source A
	INT	tag	
	DINT		
	REAL		
	stringa		

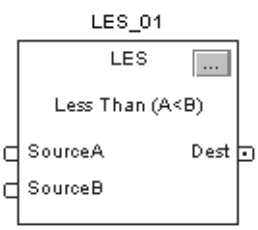
- Se si inserisce un tag SINT oppure INT, il valore viene convertito in un valore DINT con un'estensione di segno.
- I tipi di dati stringa sono:
  - tipo dati STRINGA di default
- qualunque nuovo tipo di dati stringa che si crea
- Per verificare i caratteri di una stringa, immettere un tag stringa per Source A e Source B.

Testo strutturato

```
IF sourceA < sourceB THEN
    <istruzioni>;
```

Usare il segno di minore "<" come operatore all'interno di un'espressione. Questa espressione valuta se *sourceA* è minore di *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
tag LES	FBD_COMPARE	struttura	struttura LES

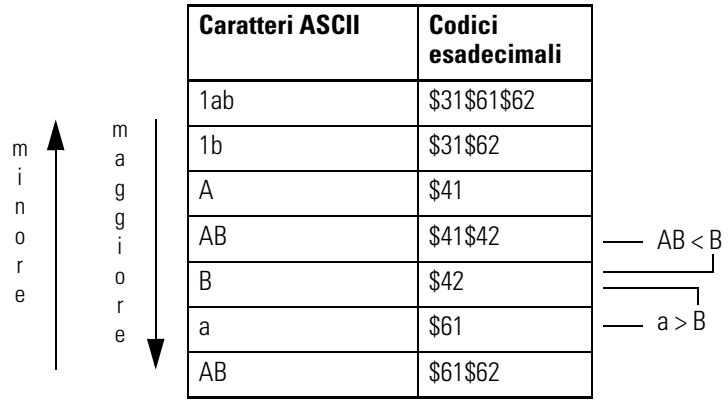
Struttura FBD_ COMPARE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	REAL	Valore da confrontare con SourceB. Valori validi = qualsiasi numero a virgola mobile
SourceB	REAL	Valore da confrontare con SourceA. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	BOOL	Risultato dell'istruzione. Equivale alla condizione del ramo di uscita dell'istruzione ladder LES.

Descrizione: L'istruzione LES verifica se Source A è minore di Source B.

Quando si confrontano stringhe:

- I valori esadecimali dei caratteri determinano se una stringa è minore o maggiore di un'altra stringa. Per il codice esadecimale di un carattere, vedere il retro copertina di questo manuale.
- Quando le due stringhe sono ordinate come in un elenco telefonico, l'ordine delle stringhe determina quale è la maggiore.



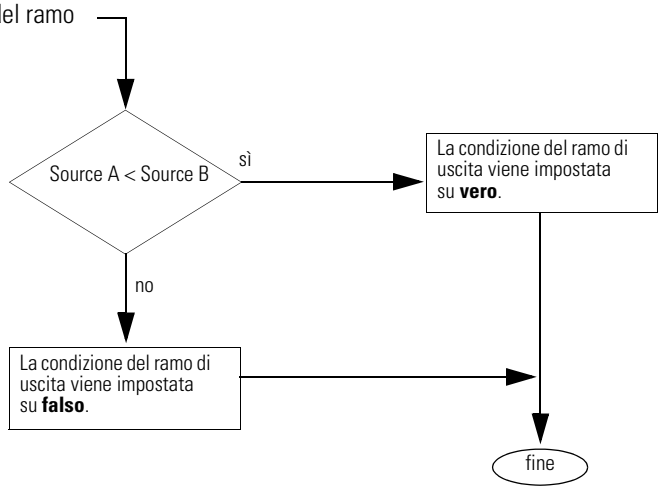
Caratteri ASCII	Codici esadecimali
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
AB	$61$62

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
quando EnableIn è falso	EnableOut viene azzerato.
quando EnableIn è vero	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

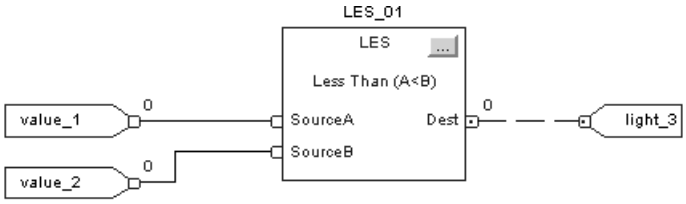
Esempio: Se *valore_1* è minore di *valore_2*, impostare *light_3*. Se *valore_1* è maggiore o uguale a *valore_2*, azzerare *light_3*.

Ladder

Testo strutturato

```
light_3 := (valore_1 < valore_2);
```

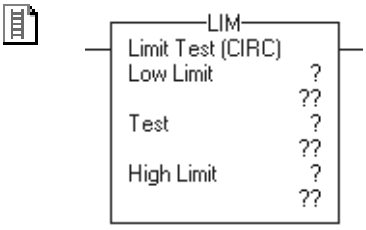
Blocco funzione



Limit (LIM)

L'istruzione LIM verifica se il valore Test è all'interno dell'intervallo compreso tra il limite inferiore ed il limite superiore.

Operandi:



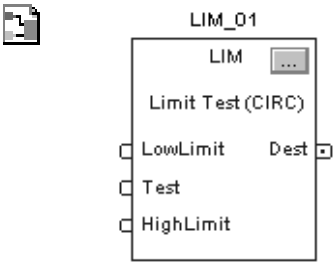
Ladder

Operando	Tipo	Formato	Descrizione
Low limit	SINT	immediato	il valore del limite inferiore
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Test	SINT	immediato	il valore da confrontare
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
High limit	SINT	immediato	il valore del limite superiore
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		

Testo strutturato

Il testo strutturato non ha un'istruzione LIM, ma consente comunque di ottenere gli stessi risultati.

```
IF (LowLimit <= HighLimit AND
    (Test >= LowLimit AND Test <= HighLimit)) OR
    (LowLimit >= HighLimit AND
    (Test <= LowLimit OR Test >= HighLimit)) THEN

          <istruzione>;

END_IF;
```



Blocco funzione

Operando	Tipo	Formato	Descrizione
tag LIM	FBD_LIMIT	struttura	struttura LIM

Struttura FBD_ LIMIT

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita come descritto sotto Esecuzione. Il valore di default è impostato.
LowLimit	REAL	Valore del limite inferiore. Valori validi = qualsiasi numero a virgola mobile
Test	REAL	Valore da confrontare con i limiti. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	BOOL	Risultato dell'istruzione. Equivale alla condizione del ramo di uscita dell'istruzione ladder LIM.
HighLimit	REAL	Valore del limite superiore. Valori validi = qualsiasi numero a virgola mobile

Descrizione: L'istruzione LIM verifica se il valore Test è all'interno dell'intervallo compreso tra il limite inferiore ed il limite superiore.

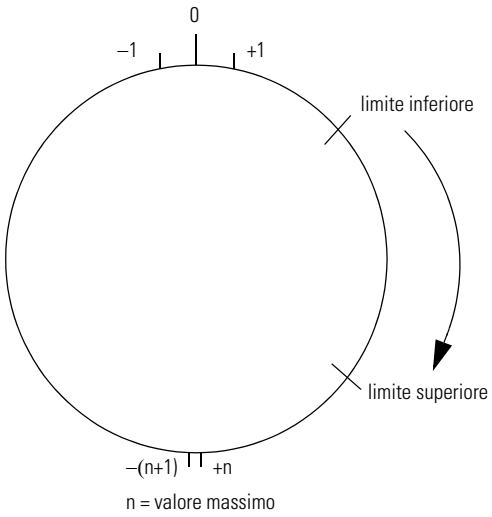
Se il limite inferiore	E il valore test è	La condizione del ramo di uscita è
≤ Limite superiore	uguale ai limiti o entro di questi	vero
	diverso dai limiti o al di fuori di questi	falso
≥ Limite superiore	uguale ai limiti o al di fuori di questi	vero
	diverso dai limiti o entro questi	falso

Gli interi con segno, quando il bit più significativo è ad 1, passano dal massimo positivo al massimo negativo. Per esempio, negli interi a 16 bit (tipo INT), il numero intero positivo massimo è 32 767, rappresentato in forma esadecimale come 16#7FFF (i bit da 0 a 14 sono tutti ad 1). Se questo numero aumenta di una unità, il risultato sarà di 16#8000 (il bit 15 è impostato). Nel caso di interi con segno, il numero esadecimale 16#8000 è uguale a –32 768 decimale. Se si continua ad incrementare fino a quando tutti i 16 bit sono ad 1, si arriva a 16#FFFF, che è pari a –1 decimale.

Questo processo può essere rappresentato come un diagramma circolare (vedere le figure seguenti). L'istruzione LIM inizia dal limite inferiore e cresce in senso orario fino a raggiungere il limite superiore. Qualsiasi valore di Test nell'intervallo in senso orario compreso tra il limite inferiore ed il limite superiore imposta la condizione del ramo di uscita a vera. Qualsiasi valore di Test nell'intervallo in senso orario compreso tra il limite superiore ed il limite inferiore imposta la condizione del ramo di uscita su falso.

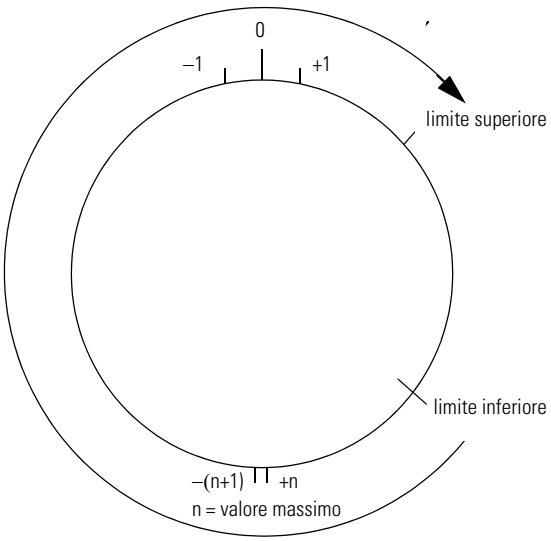
Limite inferiore ≤ Limite superiore

L'istruzione è vera se il valore Test è uguale a oppure è compreso tra il limite inferiore e superiore



Limite inferiore ≥ Limite superiore

L'istruzione è vera se il valore Test è uguale a oppure è esterno al limite inferiore e superiore



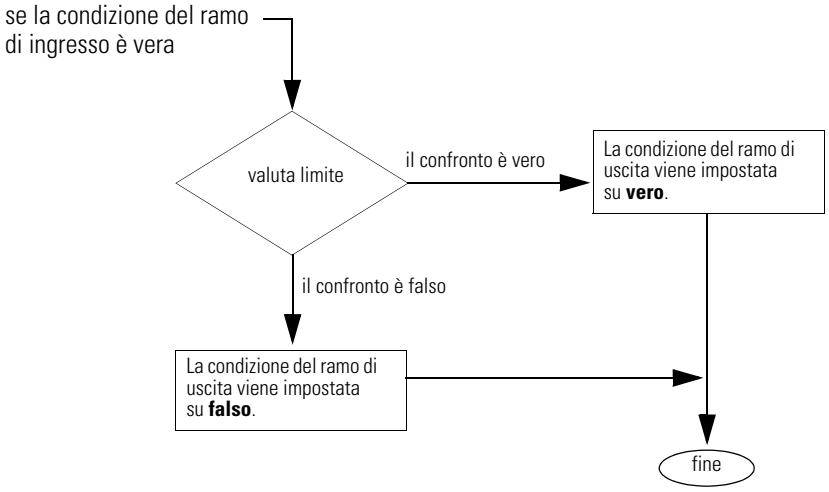
Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.


se la condizione del ramo di ingresso è vera
valuta limite
il confronto è vero
La condizione del ramo di uscita viene impostata su **vero**.
il confronto è falso
La condizione del ramo di uscita viene impostata su **falso**.
fine

post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

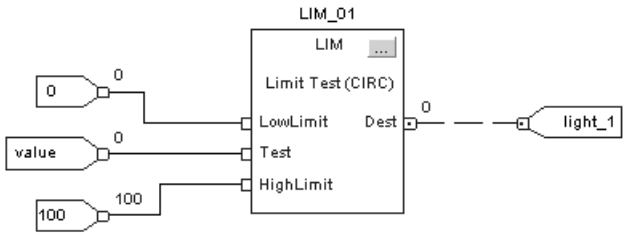
Esempio 1: **Limite inferiore ≤ Limite superiore:**
Quando 0 ≤ *valore* ≥ 100, impostare *light_1*. Se *valore* < 0 o *valore* >100, azzerare *light_1*.

Ladder

Testo strutturato

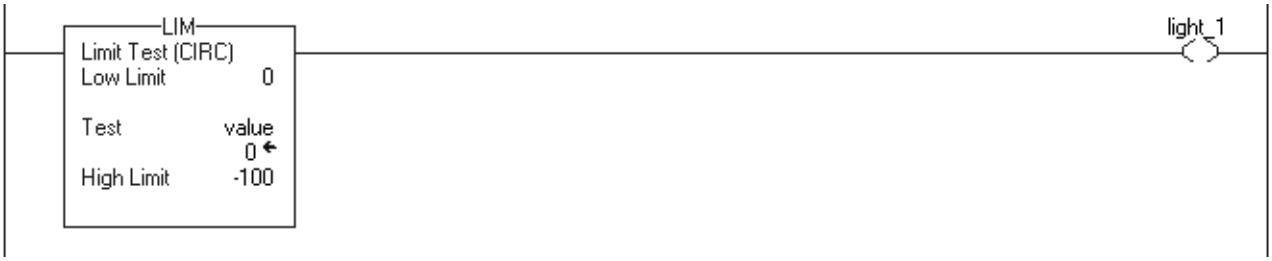
```
IF (valore <= 100 AND(valore >= 0 AND valore <= 100)) OR
   valore >= 100 AND valore <= 0 OR valore >= 100)) THEN

         light_1 := 1;

ELSE

         light_1 := 0;

END_IF;
```

Blocco funzione



Esempio 2: **Limite inferiore ≥ Limite superiore:**
Quando *valore* ≥ 0 o *valore* ≤ –100, impostare *light_1*. Se *valore* < 0 o *valore* > –100, azzerare *light_1*.

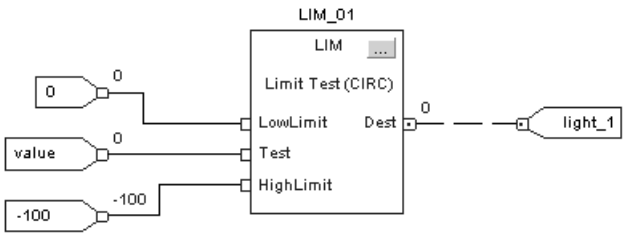
Ladder



Testo strutturato

```
IF (0 <= –100 AND valore >= 0 AND valore <= –100)) OR
    (0 >= –100 AND(valore <= 0 OR valore >= –100)) THEN

          light_1 := 1;

ELSE

          light_1 := 0;

END_IF;
```

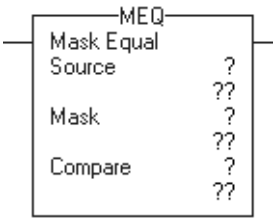
Blocco funzione



Mask Equal to (MEQ)

L'istruzione MEQ fa passare i valori di Source e Compare attraverso una maschera (Mask) e confronta i risultati.

Operandi:



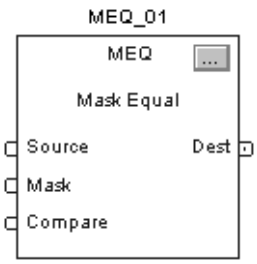
Ladder

Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	valore da confrontare con Compare
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Mask	SINT	immediato	definisce quali bit bloccare o fare passare
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Compare	SINT	immediato	valore da confrontare con Source
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		

Testo strutturato

Il testo strutturato non ha un'istruzione MEQ, ma consente comunque di ottenere gli stessi risultati.

```
IF (Source AND Mask) = (Compare AND Mask) THEN

    <istruzione>;

END_IF;
```



Blocco funzione

Operando	Tipo	Formato	Descrizione
tag MEQ	FBD_MASK_EQUAL	struttura	struttura MEQ

Struttura FBD_MASK_ EQUAL

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se è impostato, l'istruzione viene eseguita come descritto sotto Esecuzione. Il valore di default è impostato.
Source	DINT	Valore da confrontare con Confronta Valori validi = qualsiasi numero intero
Mask	DINT	Definisce quali bit bloccare (mascherare). Valori validi = qualsiasi numero intero
Compare	DINT	Valore di confronto. Valori validi = qualsiasi numero intero
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	BOOL	Risultato dell'istruzione. Equivale alla condizione del ramo di uscita dell'istruzione ladder MEQ.

Descrizione: Un "1" nella mask significa che il bit dati viene fatto passare. Uno "0" nella mask significa che il bit dati è bloccato. In genere i valori Source, Mask e Compare sono tutti dello stesso tipo di dati.

Se si mischiano tipi di dati interi, l'istruzione riempie con degli 0 i bit superiori dei tipi di dati interi più piccoli in modo che questi abbiano la stessa dimensione dei tipi di dati più grandi.

Immissione di un valore maschera immediato

Quando si immette un valore maschera, il software di programmazione imposta automaticamente valori decimali. Se si desidera immettere una maschera utilizzando un altro formato, fare precedere il prefisso corretto al valore.

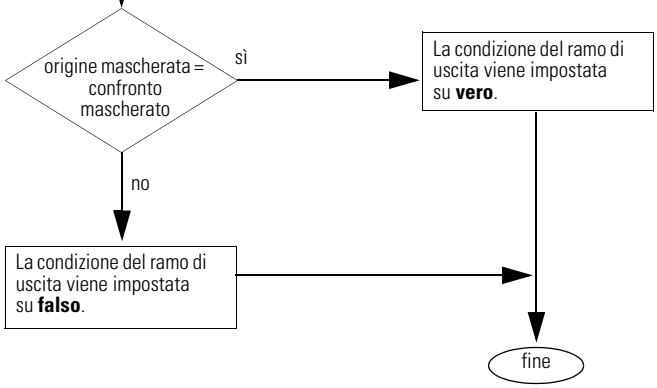
Prefisso	Descrizione
16#	Esadecimale per esempio; 16#0F0F
8#	Ottale per esempio; 8#16
2#	Binario per esempio; 2#00110011

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

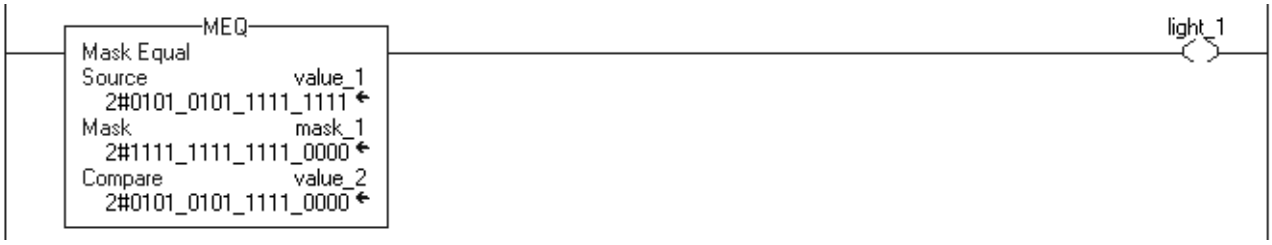
Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio 1: Se il *valore_1* mascherato è uguale al *valore_2* mascherato, impostare *light_1*. Se il *valore_1* mascherato è diverso dal *valore_2* mascherato, azzerare *light_1*. Questo esempio mostra valori mascherati uguali. Uno 0 nella mask impedisce all'istruzione di confrontare quel bit (indicato dalla x nell'esempio).

valore_1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1
mask_1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
valore_1 mascherato	0	1	0	1	0	1	0	1	1	1	1	1	X	X	X	X

valore_2	0	1	0	1	0	1	0	1	1	1	1	1	0	0	0	0
mask_1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
valore_2 mascherato	0	1	0	1	0	1	0	1	1	1	1	1	X	X	X	X

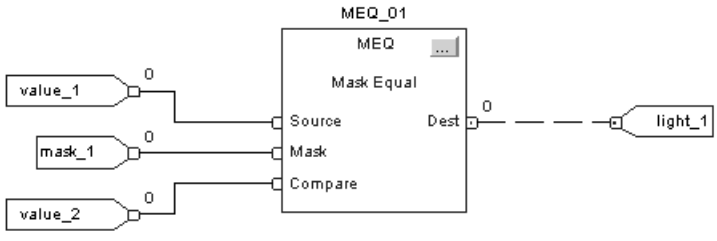
Ladder



Testo strutturato

```
light_1 := ((valore_1 AND mask_1)=(valore_2 AND
mask_2));
```

Blocco funzione



Esempio 2: Se il *valore_1* mascherato è uguale al *valore_2* mascherato, impostare *light_1*. Se il *valore_1* mascherato è diverso dal *valore_2* mascherato, azzerare *light_1*. Questo esempio mostra valori mascherati diversi. Uno 0 nella mask impedisce all'istruzione di confrontare quel bit (indicato dalla x nell'esempio).

valore_1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1
mask_1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
valore_1 mascherato	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	1

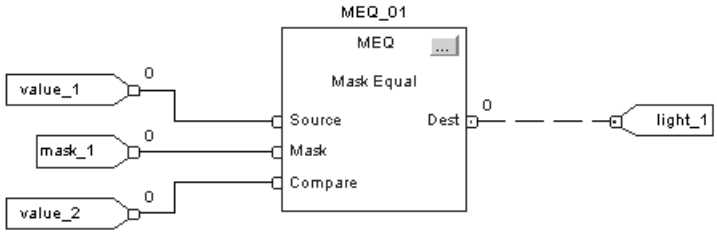
valore_2	0	1	0	1	0	1	0	1	1	1	1	1	0	0	0	0
mask_1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
valore_2 mascherato	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0

Ladder

Testo strutturato

```
light_1 := ((valore_1 AND mask_1)=(valore_2 AND
mask_2));
```

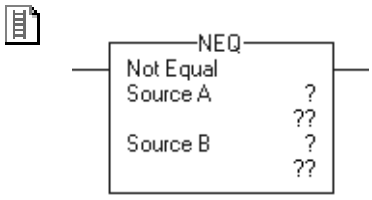
Blocco funzione



Not Equal to (NEQ)

L'istruzione NEQ verifica se Source A è diverso da Source B.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore da confrontare con Source B
	INT	tag	
	DINT		
	REAL		
	stringa		
Source B	SINT	immediato	valore da confrontare con Source A
	INT	tag	
	DINT		
	REAL		
	stringa		

- Se si inserisce un tag SINT oppure INT, il valore viene convertito in un valore DINT con un'estensione di segno.
- I tipi di dati stringa sono:
  - tipo di dati STRINGA di default
  - qualunque nuovo tipo di dati stringa che si crea
- Per verificare i caratteri di una stringa, immettere un tag stringa per Source A e Source B.

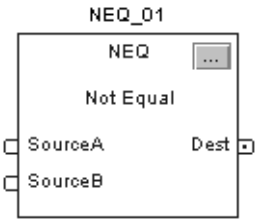
Testo strutturato

```
IF sourceA <> sourceB THEN
    <istruzioni>;
```

Usare i segni di minore e maggiore "<>" insieme come operatore all'interno di un'espressione. Questa espressione valuta se *sourceA* è diverso da *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag NEQ	FBD_COMPARE	struttura	struttura NEQ

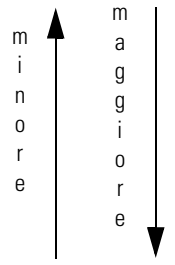
Struttura FBD_ COMPARE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	REAL	Valore da confrontare con SourceB. Valori validi = qualsiasi numero a virgola mobile
SourceB	REAL	Valore da confrontare con SourceA. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	BOOL	Risultato dell'istruzione. Equivale alla condizione del ramo di uscita dell'istruzione ladder NEQ.

Descrizione: L'istruzione NEQ verifica se Source A è diverso da Source B.

Quando si confrontano stringhe:

- Le stringhe non sono uguali se i loro caratteri non corrispondono.
- I caratteri ASCII sono sensibili alle maiuscole. "A" maiuscola ($41) *non* è uguale ad "a" minuscola ($61).



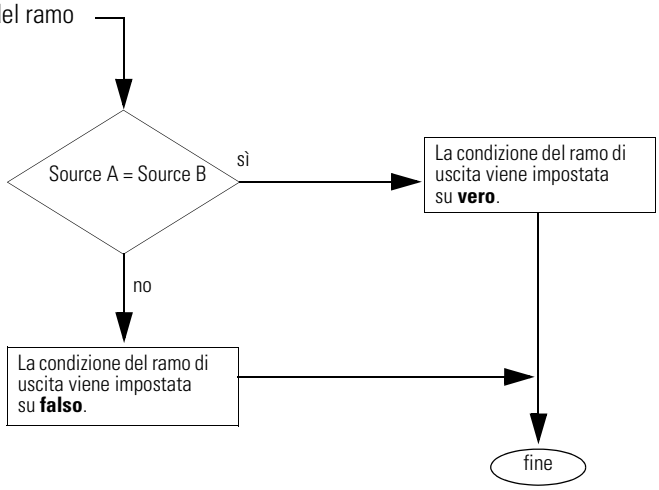
Caratteri ASCII	Codici esadecimali	
1ab	$31$61$62	
1b	$31$62	
A	$41	
AB	$41$42	AB < B
B	$42	
a	$61	a > B
AB	$61$62	

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

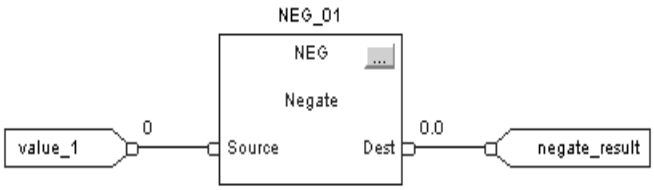
Esempio: Se *valore_1* è diverso da *valore_2*, impostare *light_4*. Se *valore_1* è uguale a *valore_2*, azzerare *light_4*.

Ladder

Testo strutturato

```
light_4 := (valore_1 <> valore_2);
```

Blocco funzione



Note:

Istruzioni di calcolo/matematiche

(CPT, ADD, SUB, MUL, DIV, MOD, SQR, SQRT, NEG, ABS)

Introduzione

Le istruzioni di calcolo/matematiche valutano le operazioni aritmetiche usando un'espressione o un'istruzione aritmetica specifica.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
valutare un'espressione	CPT	ladder testo strutturato[1]	246
sommare due valori	ADD	ladder testo strutturato[2] blocco funzione	250
sottrarre due valori	SUB	ladder testo strutturato[2] blocco funzione	253
moltiplicare due valori	MUL	ladder testo strutturato[2] blocco funzione	256
dividere due valori	DIV	ladder testo strutturato[2] blocco funzione	259
determinare il resto dopo aver diviso un valore per un altro	MOD	ladder testo strutturato[2] blocco funzione	264
calcolare la radice quadrata di un valore	SQR SQRT[3]	ladder testo strutturato blocco funzione	268
invertire il segno di un valore	NEG	ladder testo strutturato[2] blocco funzione	272
acquisire il valore assoluto di un valore	ABS	ladder testo strutturato blocco funzione	275

(1) Non esiste un'istruzione equivalente in testo strutturato. Utilizzare un'altra programmazione in testo strutturato per ottenere lo stesso risultato. Vedere la descrizione dell'istruzione.

(2) Non esiste un'istruzione equivalente in testo strutturato. Utilizzare l'operatore in un'espressione.

(3) Solo testo strutturato.

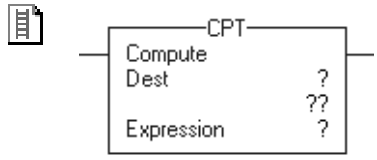
È possibile usare tipi di dati diversi, ma si potrebbe perdere in precisione ed in arrotondamento ed inoltre l'istruzione potrebbe impiegare più tempo per essere eseguita. Controllare il bit S:V per verificare se il risultato è stato troncato.

Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimale, in genere DINT o REAL.

Compute (CPT)

L'istruzione CPT esegue le operazioni aritmetiche definite nell'espressione.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Destination	SINT INT **DINT** **REAL**	tag	tag dove memorizzare il risultato
Espressione	SINT INT **DINT** **REAL**	immediato tag	un'espressione formata da tag e/o valori immediati separati da operatori
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		

Testo strutturato

Il testo strutturato non ha un'istruzione CPT, ma è possibile ottenere gli stessi risultati usando un'assegnazione e un'espressione.

```
destination := espressione_numerica;
```

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi di assegnazioni ed espressioni nel testo strutturato.

Descrizione:

L'istruzione CPT esegue le operazioni aritmetiche definite nell'espressione. Se abilitata, l'istruzione CPT valuta l'espressione e inserisce il risultato nella destinazione.

L'esecuzione di un'istruzione CPT è leggermente più lenta ed utilizza più memoria rispetto all'esecuzione di altre istruzioni di calcolo/matematiche. Il vantaggio di un'istruzione CPT consiste nel fatto che consente di inserire espressioni complesse in una istruzione.

INFORMAZIONE Non vi è *alcun limite* alla lunghezza di un'espressione.

Indicatori di stato aritmetico:

gli indicatori di stato aritmetici sono influenzati.

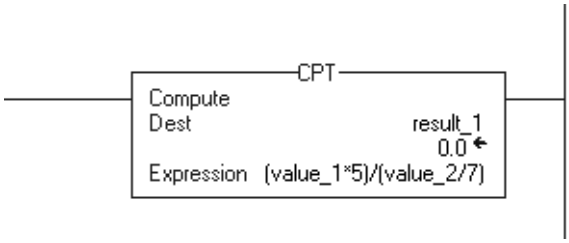
Condizioni di errore:

nessuna

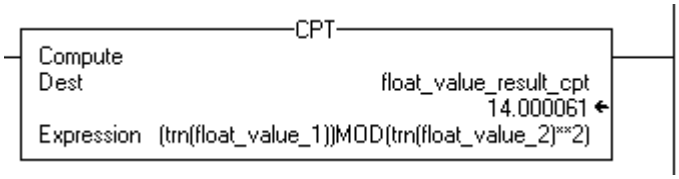
Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione valuta l'espressione e inserisce il risultato nella destinazione. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio 1: Se abilitata, l'istruzione CPT valuta *value_1* moltiplicato 5 e divide il risultato per il risultato di *value_2* diviso 7 e inserisce il risultato finale in *result_1*.



Esempio 2: Se abilitata, l'istruzione CPT tronca *float_value_1* e *float_value_2*, eleva il *float_value_2* troncato alla seconda divide il *float_value_1* troncato per il risultato e memorizza il resto della divisione in *float_value_result_cpt*.



Operatori validi

Operatore	Descrizione	Ottimale
+	somma	DINT, REAL
–	sottrai/nega	DINT, REAL
*	moltiplica	DINT, REAL
/	dividi	DINT, REAL
**	esponente (x alla y)	DINT, REAL
ABS	valore assoluto	DINT, REAL
ACS	arcocoseno	REAL
AND	AND di bit	DINT
ASN	arcoseno	REAL
ATN	arcotangente	REAL
COS	coseno	REAL
DEG	da radianti a gradi	DINT, REAL
FRD	da BCD a intero	DINT
LN	logaritmo naturale	REAL

Operatore	Descrizione	Ottimale
LOG	logaritmo in base 10	REAL
MOD	modulo-divide	DINT, REAL
NOT	complemento di bit	DINT
OR	OR di bit	DINT
RAD	da gradi a radianti	DINT, REAL
SIN	seno	REAL
SQR	radice quadrata	DINT, REAL
TAN	tangente	REAL
TOD	da intero a BCD	DINT
TRN	tronca	DINT, REAL
XOR	OR di bit esclusivo	DINT

Formattazione delle espressioni

Per ogni operatore utilizzato in un'espressione, è necessario fornire uno o due operandi (tag o valori immediati). Utilizzare la seguente tabella per formare operatori ed operandi in un'espressione:

Per operatori con:	Utilizzare questo Formato	Esempi:
un operando	operatore(operando)	ABS(*tag_a*)
due operandi	operando_a operatore operando_b	• *tag_b* + 5 • *tag_c* AND *tag_d* • (*tag_e* ** 2) MOD (*tag_f*/*tag_g*)

Determinazione dell'ordine delle operazioni

Le operazioni che vengono scritte nell'espressione sono eseguite dall'istruzione in un determinato ordine, non necessariamente nell'ordine con cui sono state scritte. È possibile cambiare l'ordine delle operazioni raggruppando i termini all'interno di parentesi, forzando l'istruzione a eseguire un'operazione all'interno delle parentesi prima di altre operazioni.

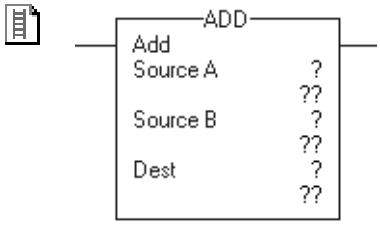
Operazioni di ordine uguale vengono eseguite da sinistra a destra.

Ordine	Funzionamento
1.	( )
2.	ABS, ACS, ASN, ATN, COS, DEG, FRD, LN, LOG, RAD, SIN, SQR, TAN, TOD, TRN
3.	**
4.	– (nega), NOT
5.	*, /, MOD
6.	– (sottrai), +
7.	AND
8.	XOR
9.	OR

Add (ADD)

L'istruzione ADD somma Source A a Source B e inserisce il risultato nella destinazione.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore da aggiungere a Source B
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Source B	SINT	immediato	valore da aggiungere a Source A
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

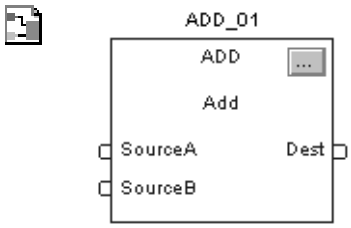
Testo strutturato

```
dest := sourceA + sourceB;
```

Usare il segno più "+" come operatore all'interno di un'espressione. Questa espressione aggiunge *sourceA* a *sourceB* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag ADD	FBD_MATH	struttura	struttura ADD

Struttura FBD_MATH

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	REAL	Valore da aggiungere a SourceB. Valori validi = qualsiasi numero a virgola mobile
SourceB	REAL	valore da aggiungere a SourceA. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: L'istruzione ADD aggiunge Source A a Source B e inserisce il risultato nella destinazione.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

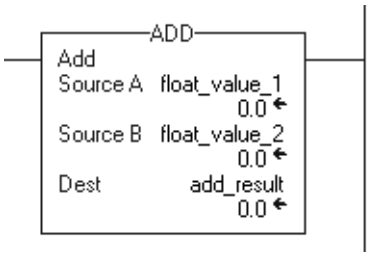
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Destination = Source A + Source B La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Aggiungere *float_value_1* a *float_value_2* e inserire il risultato in *add_result*.

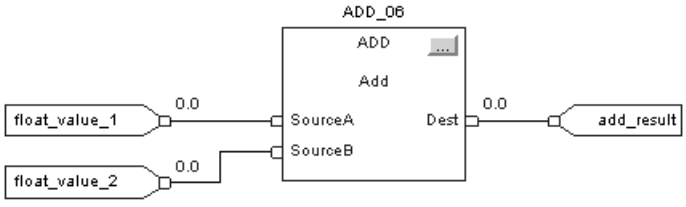
Ladder



Testo strutturato

```
add_result := float_value_1 + float_value_2;
```

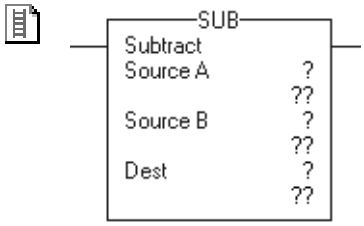
Blocco funzione



Subtract (SUB)

L'istruzione SUB sottrae Source B da Source A e inserisce il risultato nella destinazione.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore dal quale sottrarre Source B
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Source B	SINT	immediato	valore da sottrarre da Source A
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

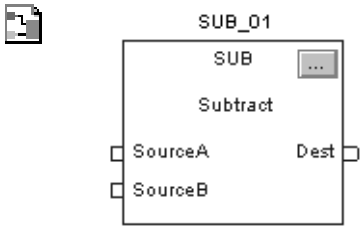
Testo strutturato

```
dest := sourceA - sourceB;
```

Usare il segno meno "–" come operatore in un'espressione. Questa espressione sottrae *sourceB* da *sourceA* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag SUB	FBD_MATH	struttura	struttura SUB

Struttura FBD_MATH

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	REAL	Valore dal quale sottrarre SourceB. Valori validi = qualsiasi numero a virgola mobile
SourceB	REAL	Valore da sottrarre da SourceA. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: L'istruzione SUB sottrae Source B da Source A e inserisce il risultato nella destinazione.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

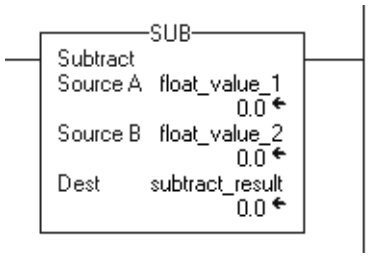
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Destination = Source B – Source A La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Sottrarre *float_value_2* da *float_value_1* e inserire il risultato in *subtract_result*.

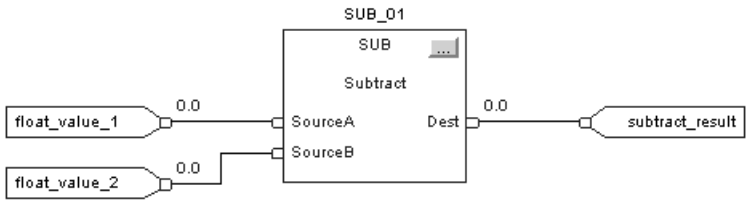
Ladder



Testo strutturato

```
subtract_result := float_value_1 – float_value_2;
```

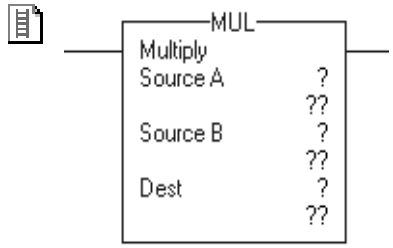
Blocco funzione



Multiply (MUL)

L'istruzione MUL moltiplica Source A per Source B e inserire il risultato nella destinazione.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore del moltiplicando
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Source B	SINT	immediato	valore del moltiplicatore
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

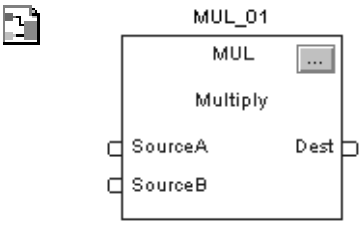
Testo strutturato

```
dest := sourceA * sourceB;
```

Usare il segno di moltiplicazione "*" come operatore in un'espressione. Questa espressione moltiplica *sourceA* per *sourceB* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag MUL	FBD_MATH	struttura	struttura MUL

Struttura FBD_MATH

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source A	REAL	Valore del moltiplicando. Valori validi = qualsiasi numero a virgola mobile
Source B	REAL	Valore del moltiplicatore. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: L'istruzione MUL moltiplica Source A per Source B e inserire il risultato nella destinazione.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

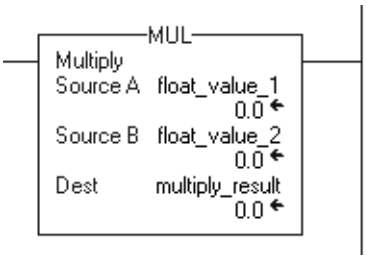
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Destination = Source B x Source A La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Moltiplicare *float_value_1* per *float_value_2* e inserire il risultato in *multiply_result*.

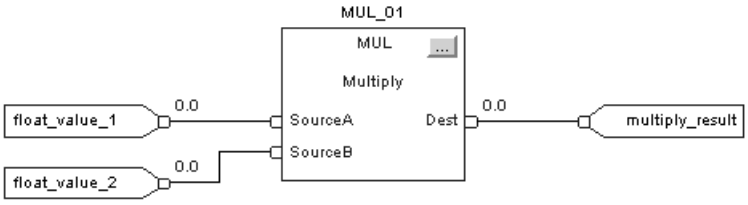
Ladder



Testo strutturato

```
multiply_result := float_value_1 * float_value_2;
```

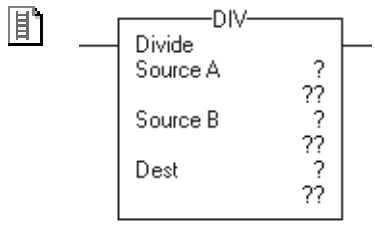
Blocco funzione



Divide (DIV)

L'istruzione DIV divide Source A per Source B e inserisce il risultato nella destinazione.

Operandi:



Ladder

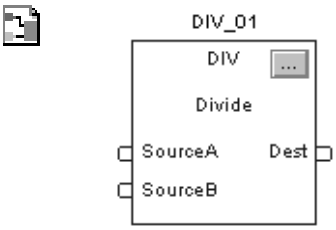
Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore del dividendo
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Source B	SINT	immediato	valore del divisore
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := sourceA/sourceB;
```

Usare il segno di divisione "/" come operatore in un'espressione. Questa espressione divide *sourceA* per *sourceB* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
tag DIV	FBD_MATH	struttura	struttura DIV

Struttura FBD_MATH

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source A	REAL	Valore del dividendo. Valori validi = qualsiasi numero a virgola mobile
Source B	REAL	Valore del divisore. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se la destinazione *non* è un valore REAL, l'istruzione gestisce la parte frazionaria del risultato come segue:

Se Source A	La parte frazionaria del risultato	Esempio		
e Source B *non* sono valori REAL	viene troncata	Source A	DINT	5
		Source B	DINT	3
		Destination	DINT	1
o Source B è un valore REAL	viene arrotondata	Source A	REAL	5.0
		Source B	DINT	3
		Destination	DINT	2

Se Source B (il divisore) è zero:

- si verifica un errore minore:
  - Tipo 4: errore programma
  - Codice 4: overflow aritmetico
- la destinazione viene impostata come segue:

Se Source B è zero e	E la destinazione è	E Il risultato è	La destinazione viene impostata a
tutti gli operandi sono interi (SINT, INT o DINT)	→	→	Source A
almeno un operando è REAL	SINT, INT o DINT	positivo	–1
		negativo	0
	REAL	positivo	1.$ (positivo infinito)
		negativo	–1.$ (negativo infinito)

Per rilevare eventuali divisioni per zero, esaminare il bit di errore minore (S:MINOR). Vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

Si verifica un errore minore se	Tipo errore	Codice errore
il divisore è zero	4	4

Esecuzione:

Ladder

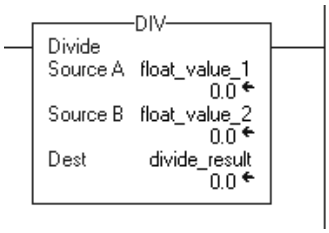
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Destination = Source A/Source B La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio 1: Dividere *float_value_1* per *float_value_2* e inserire il risultato in *divide_result*.

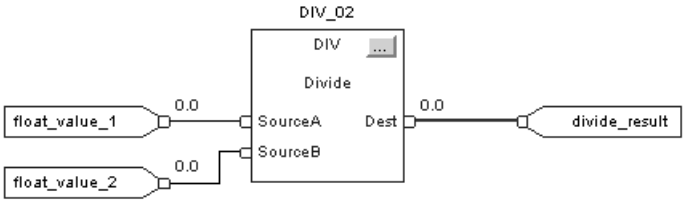
Ladder



Testo strutturato

```
divide_result := float_value_1 / float_value_2;
```

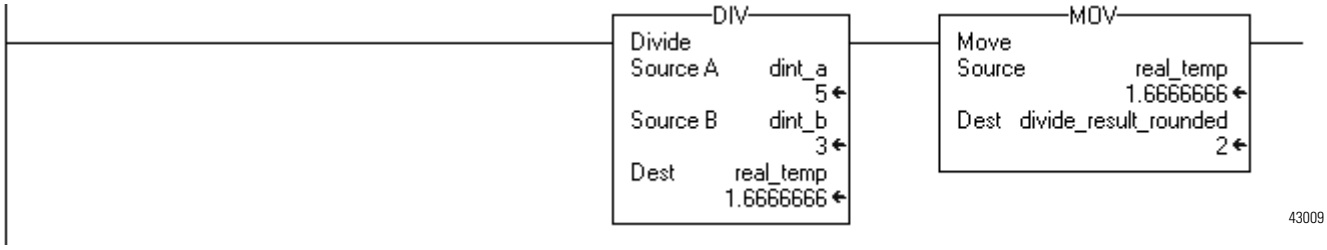
Blocco funzione



Esempio 2: Le istruzioni DIV e MOV operano insieme per dividere due interi, arrotondare il risultato e inserirlo in un tag intero:

- L'istruzione DIV divide *dint_a* per *dint_b*.
- Per arrotondare il risultato, la destinazione è un tag REAL. Se la destinazione era un tag intero (SINT, INT o DINT), l'istruzione avredde troncato il risultato.
- L'istruzione MOV sposta il risultato arrotondato (*real_temp*) da DIV a *divide_result_rounded*.
- Poiché *divide_result_rounded* è un tag DINT, il valore di *real_temp* viene arrotondato e inserito nella destinazione DINT.

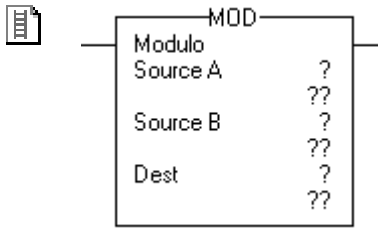
Ladder



Modulo (MOD)

L'istruzione MOD divide Source A per Source B e inserisce il resto nella destinazione.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore del dividendo
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Source B	SINT	immediato	valore del divisore
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

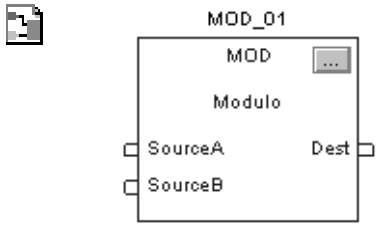
Testo strutturato

```
dest := sourceA MOD sourceB;
```

Usare MOD come operatore in un'espressione. Questa espressione divide *sourceA* per *sourceB* e memorizza il resto in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag MOD	FBD_MATH	struttura	struttura MOD

Struttura FBD_MATH

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source A	REAL	Valore del dividendo. Valori validi = qualsiasi numero a virgola mobile
Source B	REAL	Valore del divisore. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se Source B (il divisore) è zero:

- si verifica un errore minore:
  - Tipo 4: errore programma
  - Codice 4: overflow aritmetico
- la destinazione viene impostata come segue:

Se Source B è Zero e	E la destinazione è	E Il risultato è	La destinazione viene impostata a
tutti gli operandi sono interi (SINT, INT o DINT)	→	→	Source A
almeno un operando è REAL	SINT, INT o DINT	positivo	–1
		negativo	0
	REAL	positivo	1.$ (positivo infinito)
		negativo	–1.$ (negativo infinito)

Per rilevare eventuali divisioni per zero, esaminare il bit di errore minore (S:MINOR). Vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

Si verifica un errore minore se	Tipo errore	Codice errore
il divisore è zero	4	4

Esecuzione:

Ladder

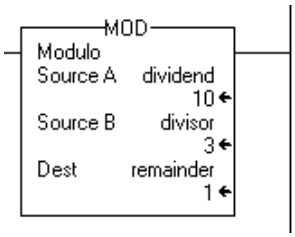
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Destination = Source A – (TRN (Source A/Source B) * Source B) La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
post-scansione	Nessuna.

Esempio: Dividere *dividend* per *divisor* e inserire il resto in *remainder*. In questo esempio, il 3 sta nel 10 tre volte, con un resto di 1.

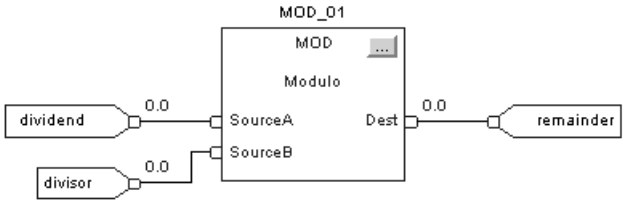
Ladder



Testo strutturato

```
remainder := dividend MOD divisor;
```

Blocco funzione

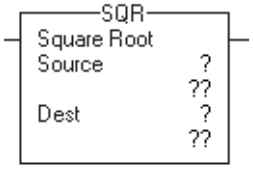


Square Root (SQR)

L'istruzione SQR calcola la radice quadrata di Source e inserisce il risultato nella destinazione.

Operandi:

Ladder



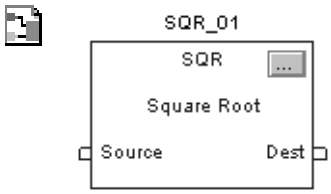
Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcolare la radice quadrata di questo valore
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := SQRT(source);
```

Usare SQRT come funzione. Questa espressione calcola la radice quadrata di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
tagSQR	FBD_MATH_ADVANCED	struttura	struttura SQR

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Calcolare la radice quadrata di questo valore. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se la destinazione *non* è un valore REAL, l'istruzione gestisce la parte frazionaria del risultato come segue:

Se il valore Source è	La parte frazionaria del risultato	Esempio		
non REAL	viene troncata	Source	DINT	3
		Destination	DINT	1
REAL	viene arrotondata	Source	REAL	3.0
		Destination	DINT	2

Se Source è negativo, l'istruzione prende il valore assoluto di Source prima di calcolare la radice quadrata.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

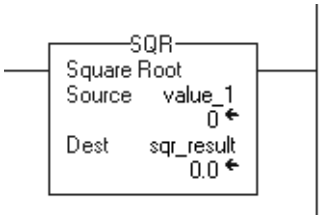
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	$Destination = \sqrt{Source}$ La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcolare la radice quadrata di *value_1* e inserire il risultato in *sqr_result*.

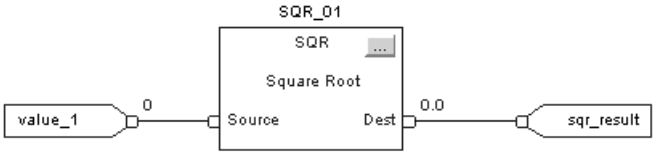
Ladder



Testo strutturato

```
sqr_result := SQRT(value_1);
```

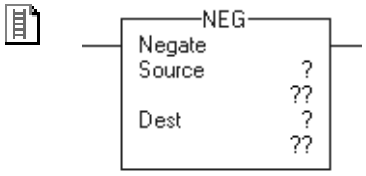
Blocco funzione



Negate (NEG)

L'istruzione NEG modifica il segno di Source e inserisce il risultato nella destinazione.

Operandi:



Ladder

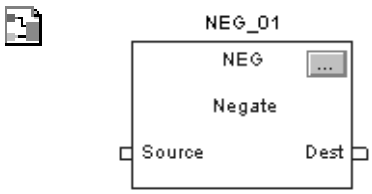
Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	valore da negare
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := -source;
```

Usare il segno meno "–" come operatore in un'espressione. Questa espressione modifica il segno di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
tag NEG	FBD_MATH_ADVANCED	struttura	struttura NEG

Struttura FBD_MATH

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Valore di cui modificare il segno. valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se si nega un valore negativo, il risultato sarà positivo. Se si nega un valore positivo, il risultato sarà negativo.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

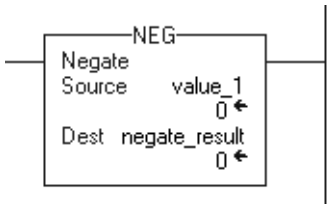
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Destination = 0 – Source La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Modificare il segno di *value_1* e inserire il risultato in *negate_result*.

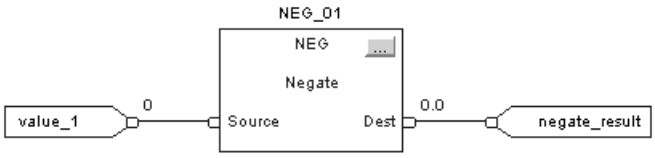
Ladder



Testo strutturato

```
negate_result := -value_1;
```

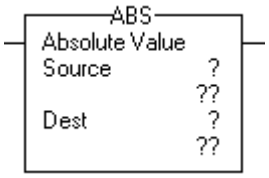
Blocco funzione



Absolute Value (ABS)

L'istruzione ABS prende il valore assoluto di Source e inserisce il risultato nella destinazione.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	valore di cui prendere il valore assoluto
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

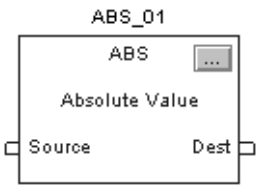
Testo strutturato

```
dest := ABS(source);
```

Usare ABS come funzione. Questa espressione calcola il valore assoluto di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
tag ABS	FBD_MATH_ADVANCED	struttura	struttura ABS

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Valore di cui prendere il valore assoluto. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: L'istruzione ABS prende il valore assoluto di Source e inserisce il risultato nella destinazione.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

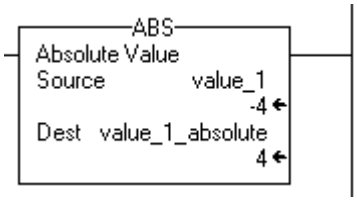
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Destination = | Source | La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Inserire il valore assoluto di *value_1* in *value_1_absolute*. In questo esempio, il valore assoluto di "–4" è "+4".

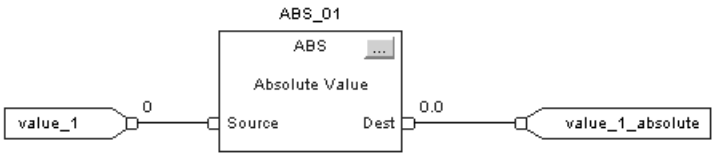
Ladder



Testo strutturato

```
value_1_absolute := ABS(value_1);
```

Blocco funzione



Note:

Istruzioni di spostamento/logiche
(MOV, MVM, BTD, MVMT, BTDT, CLR, SWPB, AND, OR, XOR, NOT, BAND, BOR, BXOR, BNOT)

Introduzione

È possibile usare tipi di dati diversi, ma si potrebbe perdere in precisione ed in arrotondamento ed inoltre l'istruzione potrebbe impiegare più tempo per essere eseguita. Controllare il bit S:V per verificare se il risultato è stato troncato.

Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimali, in genere DINT o REAL.

Le istruzioni di spostamento servono per modificare e spostare i bit.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
copiare un valore	MOV	ladder testo strutturato[1]	281
copiare una sezione specifica di un numero intero	MVM	ladder	283
copiare una sezione specifica di un numero intero in un blocco funzione	MVMT	testo strutturato blocco funzione	286
spostare i bit all'interno di un numero intero o da un numero intero all'altro	BTD	ladder	290
spostare i bit all'interno di un numero intero o da un numero intero all'altro all'interno di un blocco funzione	BTDT	testo strutturato blocco funzione	293
cancellare un valore	CLR	testo strutturato[1] ladder	296
riposizionare i byte di un tag INT, DINT, o REAL	SWPB	ladder testo strutturato	298

(1) Non esiste un'istruzione equivalente in testo strutturato. Utilizzare un'altra programmazione in testo strutturato per ottenere lo stesso risultato. Vedere la descrizione dell'istruzione.

Le istruzioni logiche eseguono delle operazioni sui bit.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
operazione di AND di bit	Bitwise AND &[1]	ladder testo strutturato[2] blocco funzione	302
operazione OR di bit	Bitwise OR	ladder testo strutturato[2] blocco funzione	305
operazione OR esclusivo di bit	Bitwise XOR	ladder testo strutturato[2] blocco funzione	308
operazione NOT di bit	Bitwise NOT	ladder testo strutturato[2] blocco funzione	311
AND logico su 8 ingressi booleani.	Boolean AND (BAND)	testo strutturato[2] blocco funzione	314
OR logico su 8 ingressi booleani.	Boolean OR (BOR)	testo strutturato[2] blocco funzione	317
esegue un OR esclusivo su 2 ingressi booleani.	Boolean Exclusive OR (BXOR)	testo strutturato[2] blocco funzione	320
fare il complemento di un ingresso booleano.	Boolenan NOT (BNOT)	testo strutturato[2] blocco funzione	323

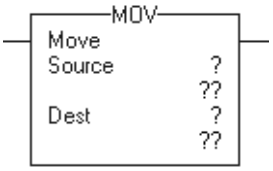
(1) Solo testo strutturato.

(2) In testo strutturato, le operazioni AND, OR, XOR e NOT possono essere di tipo di bit o logico.

Move (MOV)

L'istruzione MOV copia il valore Source nella Destination. Il valore Source rimane invariato.

Operandi:



Ladder

Operando	Tipo	formato	Descrizione
Source	SINT	immediato	valore da spostare (copiare)
	INT	tag	
	DINT		
	REAL		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := Source;
```

Utilizzare un'assegnazione ":=" con un'espressione. Con questa assegnazione il valore viene spostato dalla *source* alla *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni e delle assegnazioni in testo strutturato.

Descrizione: L'istruzione MOV copia il valore Source nella Destination. Il valore Source rimane invariato.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

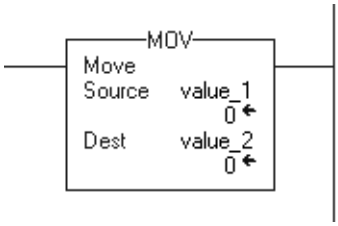
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione copia il valore Source nella Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Portare i dati di *value_1* in *value_2*.

Ladder



Testo strutturato

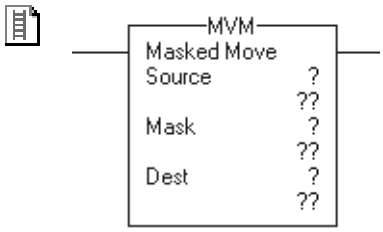
```
value_2: = value_1;
```

Masked Move (MVM)

L'istruzione MVM sposta una copia di Source in una destinazione attraverso una maschera.

L'istruzione è disponibile in testo strutturato e blocco funzione come MVMT, vedere pagina 286.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	valore da spostare
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Mask	SINT	immediato	quali bit bloccare o fare passare
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		

Testo strutturato

```
dest := (Dest AND NOT (Mask))
    OR (Source AND Mask);
```

Questa istruzione è disponibile come MVMT in testo strutturato. In alternativa è possibile combinare la logica di bit all'interno di un'espressione e assegnare il risultato alla destinazione. Questa espressione consente di eseguire uno spostamento su *Source* attraverso una maschera.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni e delle assegnazioni in testo strutturato.

Descrizione:

L'istruzione MVM utilizza una maschera per passare o bloccare i bit di dati della Source. Un "1" nella maschera significa che il bit dati viene fatto passare. Uno "0" nella maschera significa che il bit dati è bloccato.

Se si mischiano tipi di dati interi, l'istruzione riempie con degli 0 i bit superiori dei tipi di dati interi più piccoli in modo che questi abbiano la stessa dimensione dei tipi di dati più grandi.

Immissione di un valore di maschera immediato

Quando si immette una maschera, il software di programmazione imposta automaticamente valori decimali. Se si desidera immettere una maschera utilizzando un altro formato, fare precedere il prefisso corretto al valore.

Prefisso	Descrizione
16#	Esadecimale per esempio; 16#0F0F
8#	Ottale per esempio; 8#16
2#	Binario per esempio; 2#00110011

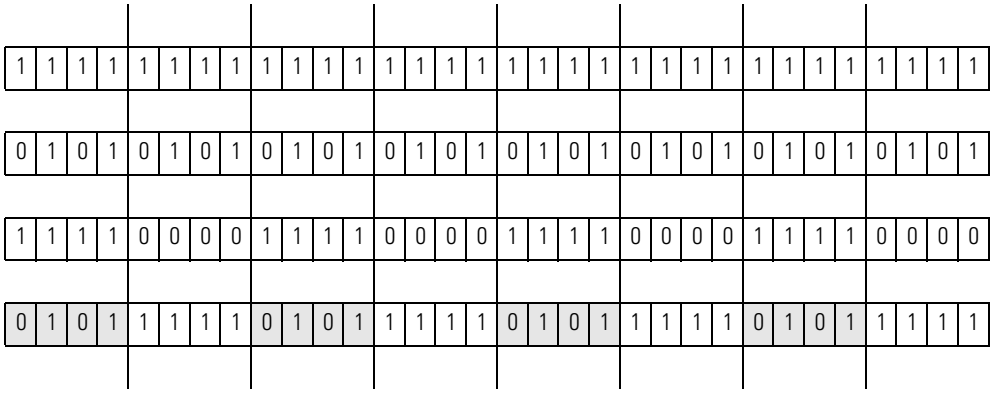
Indicatori di stato aritmetico gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore nessuna

Esecuzione:

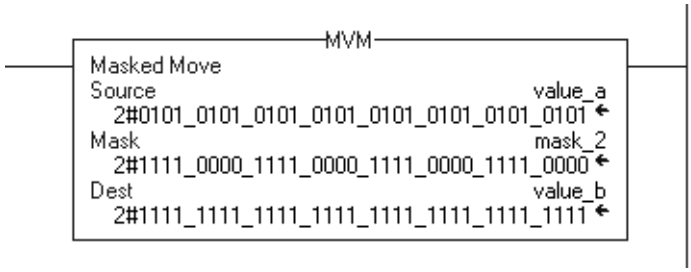
Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione fa passare i valori della Source attraverso la maschera e muove i risultati nella destinazione. I bit senza maschera della destinazione rimangono invariati. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Copia dati da *value_a* a *value_b*, consentendo di applicare una maschera (lo 0 maschera i dati in *value_a*).



Le caselle ombreggiate indicano i bit che sono stati modificati in *value_b*.

Ladder



Testo strutturato

```
value_b := (value_b AND NOT (mask_2)) OR
    (value_a AND mask_2);
```

Masked Move with Target (MVMT)

L'istruzione MVMT per prima cosa copia il valore Target nella destinazione. Quindi l'istruzione confronta Source a cui è stata applicata una maschera, con la destinazione ed apporta le eventuali modifiche richieste alla destinazione. Source e Target rimangono invariati.

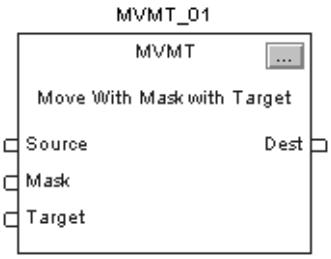
Questa istruzione è disponibile in ladder come MVM, vedere pagina 283.

Operandi:

```
MVMT(MVMT_tag);
```

Testo strutturato

Variabile	Tipo	Formato	Descrizione
tag MVMT	FBD_MASKED_MOVE	struttura	struttura MVMT



Blocco funzione

Operando	Tipo	Formato	Descrizione
tag MVMT	FBD_MASKED_MOVE	struttura	struttura MVMT

Struttura FBD_MASKED_MOVE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Blocco funzione** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato** Nessun effetto. L'istruzione viene eseguita.
Source	DINT	Il valore dell'ingresso viene portato nella destinazione sulla base del valore della maschera. Valori validi = qualsiasi numero intero
Mask	DINT	Maschera di bit da spostare da Source a dest. Tutti i bit impostati a uno provocano lo spostamento dei bit corrispondenti dalla Source alla dest. Tutti i bit impostati a zero non determinano lo spostamento dei bit corrispondenti dalla Source alla dest. Valori validi = qualsiasi numero intero
Target	DINT	Il valore dell'ingresso viene portato alla destinazione prima del passaggio dei bit Source attraverso la maschera. Valori validi = qualsiasi numero intero

Parametro di uscita	Tipo di dati	Descrizione
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Uscita dell'istruzione con spostamento attraverso maschera. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se abilitata, l'istruzione MVMT utilizza una maschera per passare o bloccare i bit di dati della Source. Un "1" nella mask significa che il bit dati viene fatto passare. Uno "0" nella mask significa che il bit dati è bloccato.

Se si mischiano tipi di dati interi, l'istruzione riempie con degli 0 i bit superiori dei tipi di dati interi più piccoli in modo che questi abbiano la stessa dimensione dei tipi di dati più grandi.

Inserimento di un valore immediato maschera con un Riferimento di ingresso

Quando si immette una masschera, il software di programmazione imposta automaticamente valori decimali. Se si desidera immettere una maschera utilizzando un altro formato, fare precedere il prefisso corretto al valore.

Prefisso	Descrizione
16#	Esadecimale per esempio; 16#0F0F
8#	Ottale per esempio; 8#16
2#	Binario per esempio; 2#00110011

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

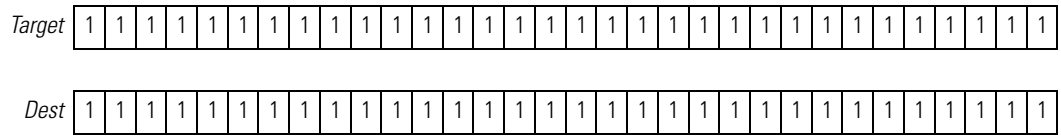
Condizioni di errore: nessuna

Esecuzione:

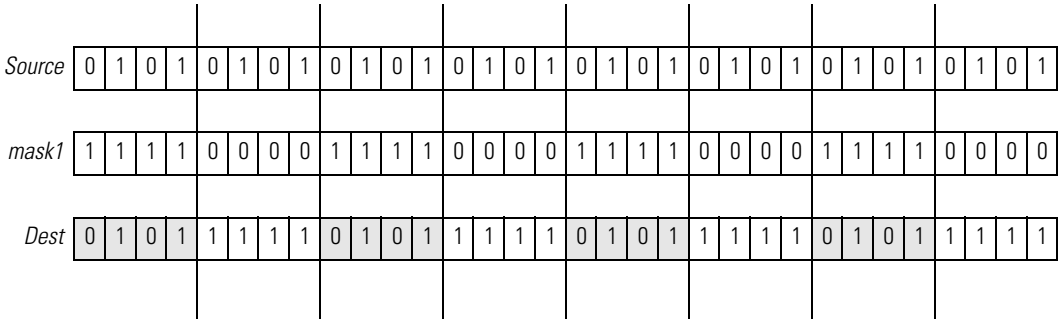
Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Nessuna.	Nessuna.
prima scansione dell'istruzione	Nessuna.	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.	na
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.	EnableIn viene sempre impostato. L'istruzione viene eseguita.
post-scansione	Nessuna.	Nessuna.

Esempio:

1. Copia Target nella Dest.



2. Applica la maschera alla Source e la confronta con la Destination. Le modifiche eventualmente necessarie vengono effettuate nella Destination. Source e Target rimangono invariati. Uno 0 nella maschera impedisce all'istruzione di confrontare quel bit (indicato dalla x nell'esempio).

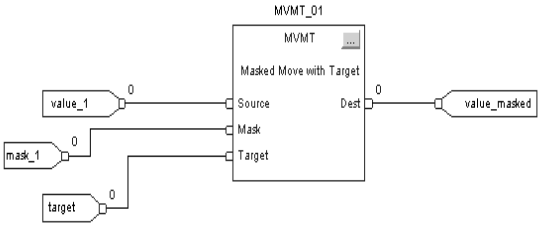


Le caselle ombreggiate indicano i bit che sono stati modificati.

Testo strutturato

```
MVMT_01.Source := value_1;
MVMT_01.Mask := mask1;
MVMT_01.Target := target;

MVMT(MVMT_01);

value_masked := MVMT_01.Dest;
```

Blocco funzione



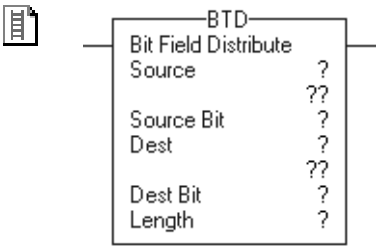
Bit Field Distribute (BTD)

L'istruzione BTD copia i bit specificati dalla Source, sposta i bit nella posizione appropriata, e scrive i bit nella Destination.

Questa istruzione è disponibile come BTDT in testo strutturato e blocco funzione, vedere pagina 293.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	tag che contiene i bit da spostare
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Source bit	DINT	immediato (0–31 DINT) (0–15 INT) (0–7 SINT)	numero del bit (bit con il numero più basso) da cui iniziare uno spostamento deve rientrare nella gamma valida per il tipo di dati della Source
Destination	SINT	tag	tag in cui spostare i bit
	INT		
	DINT		
Destination bit	DINT	immediato (0–31 DINT) (0–15 INT) (0–7 SINT)	numero del bit (bit con il numero più basso) da cui iniziare la copia dei bit dalla Source deve rientrare nella gamma valida per il tipo di dati della Destination
Length	DINT	immediato (1–32)	numero di bit da spostare

Descrizione:

Quando è abilitata, l'istruzione BTD copia un gruppo di bit dalla Source alla destinazione. Il gruppo di bit è identificato dal bit di Source (bit con il numero più basso del gruppo) e la Length (n. di bit da copiare). Il bit di destinazione è il bit con il numero inferiore da cui cominciare nella Destination. Il valore Source rimane invariato.

Se la lunghezza del campo bit è superiore alla Destination, l'istruzione non determina il salvataggio dei bit in eccesso. Gli eventuali bit in eccesso non scorrono nella parola successiva.

Se si mischiano tipi di dati interi, l'istruzione riempie con degli 0 i bit superiori dei tipi di dati interi più piccoli in modo che questi abbiano la stessa dimensione dei tipi di dati più grandi.

Indicatori di stato aritmetico:

non influenzati

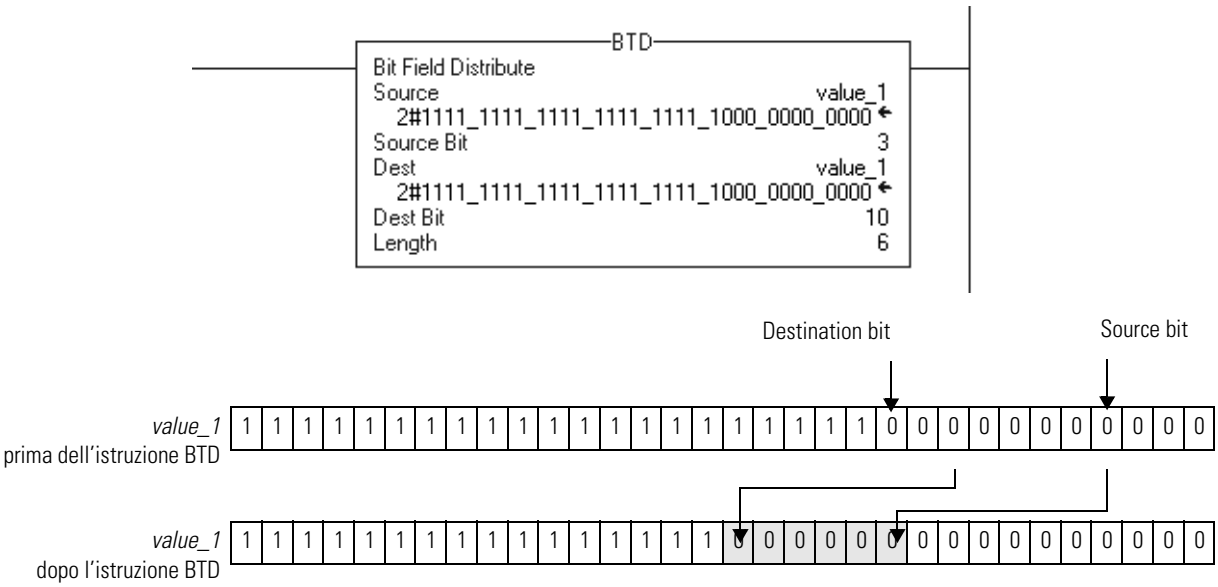
Condizioni di errore:

nessuna

Esecuzione:

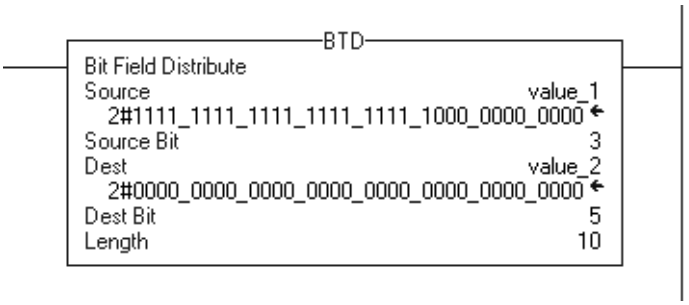
Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione copia e sposta i bit della Source nella Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

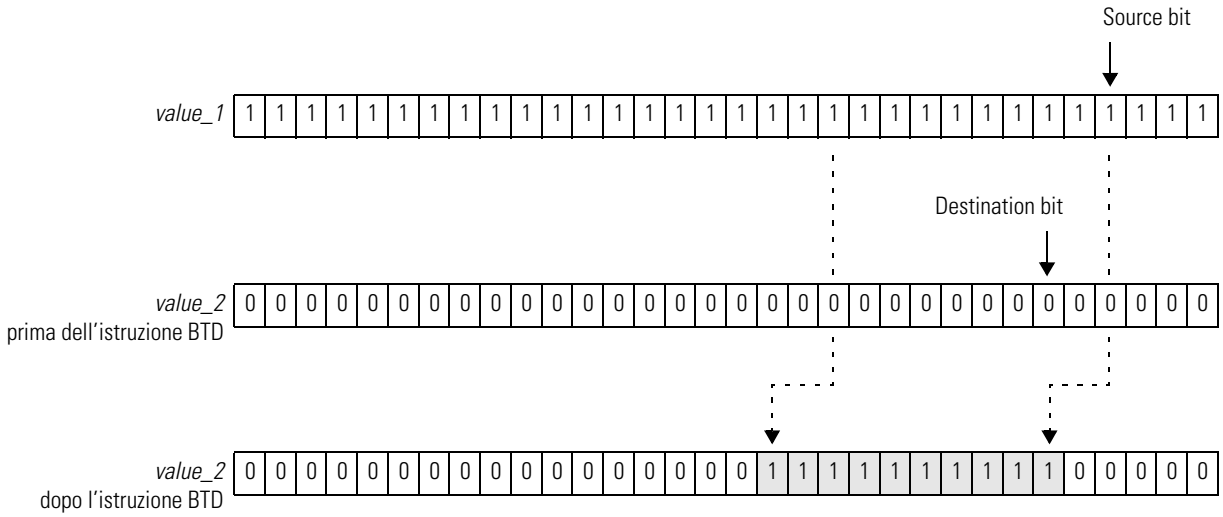
Esempio 1: Se abilitata, l'istruzione BTD determina lo spostamento dei bit all'interno del *value_1*.



Le caselle ombreggiate indicano i bit che sono stati modificati in *value_1*.

Esempio 2: Se è abilitata, l'istruzione BTD sposta 10 bit da *value_1* a *value_2*.





Le caselle ombreggiate indicano i bit che sono stati modificati in *value_2*.

Bit Field Distribute with Target (BTDT)

L'istruzione BTDT per prima cosa copia il valore Target nella Destinazione. Quindi l'istruzione copia i bit specificati da Source, sposta i bit nella posizione appropriata, e scrive i bit nella Destination. Source e Target rimangono invariati.

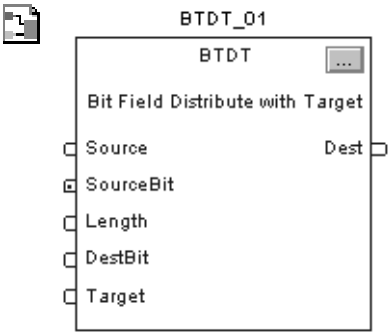
Questa istruzione è disponibile in ladder come BTD, vedere pagina 290.

Operandi:

```
BTDT(BTDT_tag);
```

Testo strutturato

Variabile	Tipo	Formato	Descrizione
BTDT tag	FBD_BIT_FIELD_DISTRIBUTE	struttura	struttura BTDT



Blocco funzione

Operando	Tipo	Formato	Descrizione
BTDT tag	FBD_BIT_FIELD_DISTRIBUTE	struttura	struttura BTDT

Struttura FBD_BIT_FIELD_DISTRIBUTE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	**Blocco funzione:** Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Se impostato, l'istruzione viene eseguita. Il valore di default è impostato. **Testo strutturato:** Nessun effetto. L'istruzione viene eseguita.
Source	DINT	Valore dell'ingresso contenente i bit da portare nella destinazione. Valori validi = qualsiasi numero intero
SourceBit	DINT	Posizione del bit nella Source (bit con il numero più basso da cui iniziare lo spostamento). Validi = 0–31
Length	DINT	numero di bit da spostare Validi = 1–32
DestBit	DINT	Posizione del bit nella Source (bit con il numero più basso da cui iniziare la copia). Validi = 0–31

Parametro di ingresso	Tipo di dati	Descrizione
Target	DINT	Il valore dell'ingresso viene portato alla destinazione prima dello spostamento dei bit dalla Source. Valori validi = qualsiasi numero intero
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Risultato dell'operazione di spostamento bit. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Quando è abilitata, l'istruzione BTD copia un gruppo di bit dalla Source alla destinazione. Il gruppo di bit è identificato dal bit di Source (bit con il numero più basso del gruppo) e la Length (n. di bit da copiare). Il bit di destinazione è il bit con il numero inferiore da cui cominciare nella Destination. Il valore Source rimane invariato.

Se la lunghezza del campo bit è superiore alla Destination, l'istruzione non determina il salvataggio dei bit in eccesso. Gli eventuali bit in eccesso non scorrono nella parola successiva.

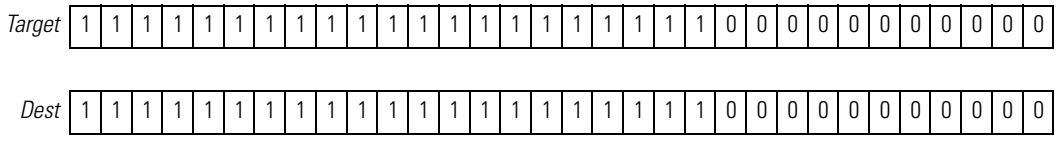
Indicatori di stato aritmetico: Gli indicatori di stato aritmetici sono influenzati

Condizioni di errore: nessuna

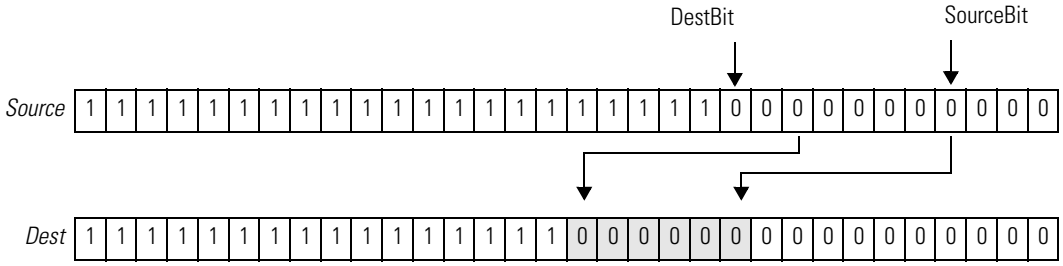
Esecuzione:

Condizione	Azione blocco funzione	Azione testo strutturato
durante la prescansione	Nessuna.	Nessuna.
prima scansione dell'istruzione	Nessuna.	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato, l'istruzione non esegue alcuna azione e le uscite non vengono aggiornate.	na
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.	EnableIn viene sempre impostato. L'istruzione viene eseguita.
post-scansione	Nessuna.	Nessuna.

Esempio: **1.** Il controllore copia il Target nella Dest.



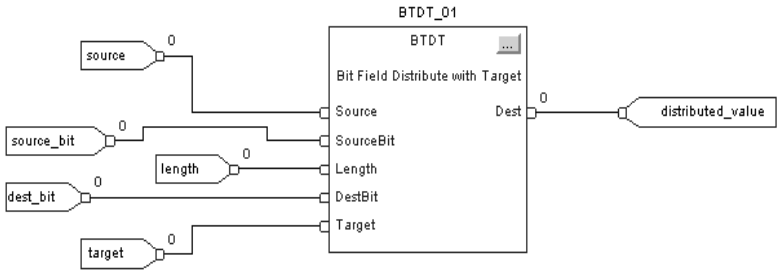
2. SourceBit e Length indicano quali bit della Source devono essere copiati nella Dest, a partire da DestBit. Source e Target rimangono invariati.



Testo strutturato

```
BTDT_01.Source := Source;
BTDT_01.SourceBit := source_bit;
BTDT_01.Length := length;
BTDT_01.DestBit := dest_bit;
BTDT_01.Target := target;

BTDT(BTDT_01);

distributed_value := BTDT_01.Dest;
```

Blocco funzione

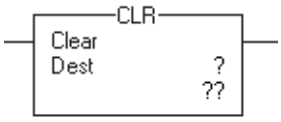


Clear (CLR)

L'istruzione CLR cancella tutti i bit della destinazione.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Destination	SINT	tag	tag da cancellare
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := 0;
```

Il testo strutturato non prevede un'istruzione CLR. In alternativa, assegnare 0 al tag da cancellare. Con questa assegnazione *dest.* viene cancellato

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni e delle assegnazioni in testo strutturato.

Descrizione:

L'istruzione CLR cancella tutti i bit della destinazione.

Indicatori di stato aritmetico:

gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

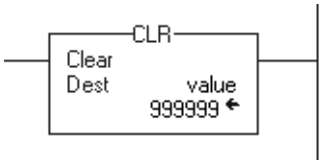
nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione cancella la Destinazione. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Cancella tutti i bit di *value*.

Ladder



Testo strutturato

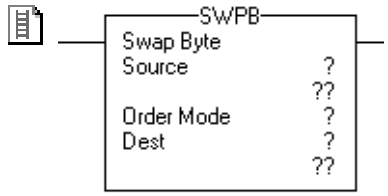
```
value := 0;
```

Swap Byte (SWPB)

L'istruzione SWPB ridispone i byte di un valore.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare		
Source	INT DINT REAL	tag	tag contenente i byte che si desidera ridisporre		
Modalità order			**Se Source è**	**E si desidera modificare la configurazione dei byte impostando questa configurazione (ogni lettera rappresenta un byte diverso)**	**Selezionare**
			INT	N/A	qualunque opzione
			DINT	ABCD ⇒ DCBA	REVERSE (o inserire 0)
			REAL	ABCD ⇒ CDAB	WORD (o inserire 1)
				ABCD ⇒ BADC	HIGH/LOW (o inserire 2)
Destination	INT DINT REAL	tag	tag per memorizzare i byte con un nuovo ordine		
			Se Source è	**La Destination deve essere**	
			INT	INT	
				DINT	
			DINT	DINT	
			REAL	REAL	

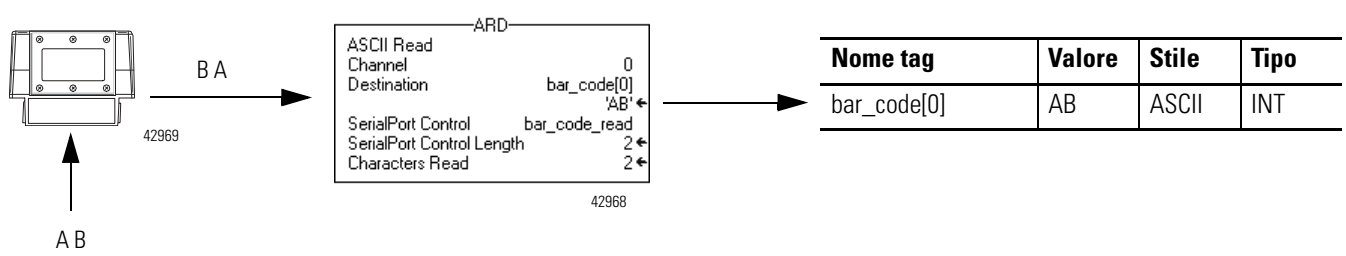
Testo strutturato

```
SWPB(Source,OrderMode,Dest);
```

Gli operandi sono gli stessi dell'istruzione SWPB in ladder. Selezionando la modalità ordine HIGH/LOW, inserire HIGHLOW o HIGH_LOW (senza barra obliqua).

Descrizione: L'istruzione SWPB consente di cambiare l'ordine dei byte della Source. I risultati vengono inseriti nella Destinazione.

Quando si leggono o scrivono caratteri ASCII normalmente *non* è necessario cambiare l'ordine dei caratteri. Le istruzioni di lettura e scrittura ASCII (ARD, ARL, AWA, AWT) determinano automaticamente lo scambio dei caratteri, come mostrato sotto.



Nome tag	Valore	Stile	Tipo
bar_code[0]	AB	ASCII	INT

Indicatori di stato aritmetico: non influenzati

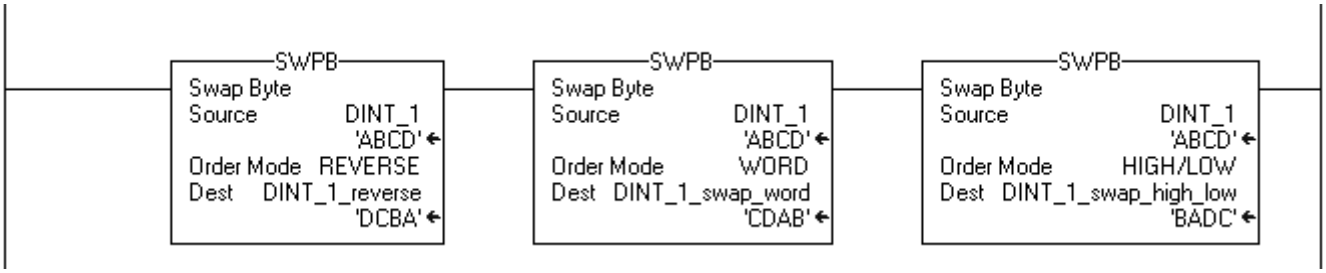
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione SWPB ridispone i byte specificati.	L'istruzione SWPB ridispone i byte specificati.
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio 1: Le tre istruzioni SWPB determinano ciascuna il riordino dei byte di *DINT_1* con un ordine diverso. Lo stile di visualizzazione è ASCII, e ciascun carattere rappresenta un byte. Ciascuna istruzione colloca il byte in una Destinazione diversa, nel nuovo ordine.

Ladder



Testo strutturato

```
SWPB(DINT_1,REVERSE,DINT_1_reverse);
SWPB(DINT_1,WORD,DINT_1_swap_word);
SWPB(DINT_1,HIGHLOW,DINT_1_swap_high_low);
```

Esempio 2: Nell'esempio che segue vengono invertiti i byte di ciascun elemento di una matrice. Per visualizzare un progetto RSLogix 5000 contenente questo esempio, aprire la cartella RSLogix 5000\Projects\Samples, file Swap_Bytes_in_Array.ACD.

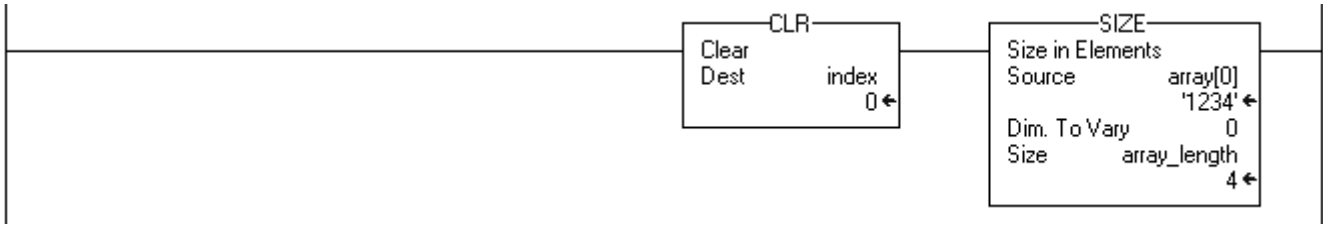
1. Inizializzare i tag. L'istruzione SIZE trova il numero di elementi all'interno della *array* e memorizza quel valore in *array_length*. Un'istruzione successiva utilizza questo valore per stabilire quando la routine ha agito su tutti gli elementi della matrice.

2. Inversione dei byte in un elemento della *array*.
   - L'istruzione SWPB inverte i byte dell' elemento in posizione indicata dal valore *index*. Ad esempio, se *index* è pari a 0, l'istruzione SWPB agisce su *array[0]*.
   - L'istruzione ADD aumenta il valore di *index*. La volta successiva che l'istruzione viene eseguita, SWPB agisce sull'elemento successivo della *array*.

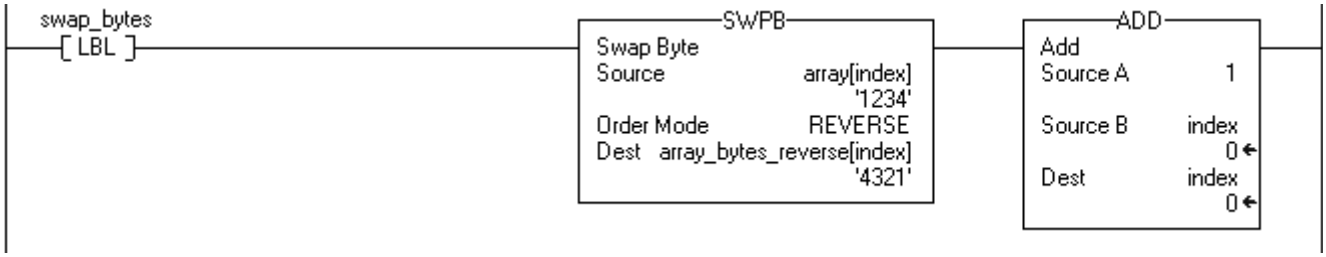
3. Stabilire quando l'istruzione SWPB ha agito su tutti gli elementi della matrice.
   - Se *index* è inferiore al numero di elementi della matrice (*array_length*), si passa all'elemento successivo della matrice.
   - Se *index* è pari a *array_length*, significa che SWPB ha agito su tutti gli elementi della matrice.

Ladder

Inizializzare i tag.



Inversione dei byte.



Stabilire se l'istruzione SWPB ha agito su tutti gli elementi della matrice.

Testo strutturato

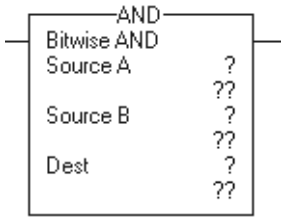
```
index := 0;
SIZE (array[0],0,array_length);
REPEAT
 SWPB(array[index],REVERSE,array_byte_reverse[index]);
 index := index + 1;
UNTIL(index >= array_length)END_REPEAT;
```

Bitwise AND (AND)

L'istruzione AND esegue un'operazione AND di bit utilizzando i bit di Source A e Source B e colloca i risultati nella Destinazione.

Per eseguire un AND logico, vedere pagina 314.

Operandi:



Ladder

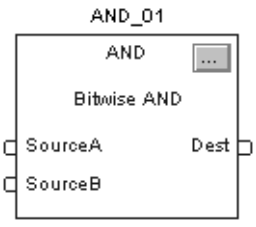
Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore su cui eseguire AND con Source B
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Source B	SINT	immediato	valore su cui eseguire AND con Source A
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Destination	SINT	tag	memorizza il risultato
	INT		
	DINT		

Testo strutturato

```
dest := sourceA AND sourceB
```

Usare AND o il simbolo della e commerciale "&" come operatore all'interno di un'espressione. Questa espressione valuta *sourceA* AND *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
AND tag	FBD_LOGICAL	struttura	struttura AND

Struttura FBD_LOGICAL

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	DINT	Valore su cui eseguire AND con Source B. Valori validi = qualsiasi numero intero
SourceB	DINT	Valore su cui eseguire AND con Source A. Valori validi = qualsiasi numero intero
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Risultato dell'istruzione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se è abilitata, l'istruzione valuta l'operazione AND.

Se il bit In Source A è	E il bit in Source B è	Il bit nella Destination è
0	0	0
0	1	0
1	0	0
1	1	1

Se si mischiano tipi di dati interi, l'istruzione riempie con degli 0 i bit superiori dei tipi di dati interi più piccoli in modo che questi abbiano la stessa dimensione dei tipi di dati più grandi.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

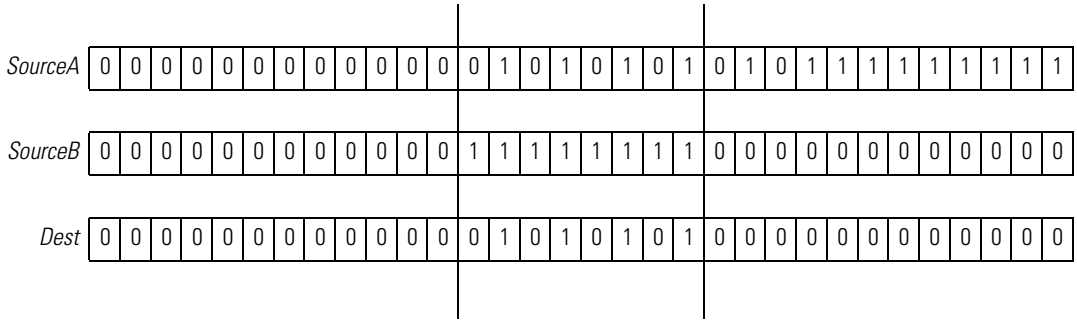
Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione esegue un'operazione AND di bit. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

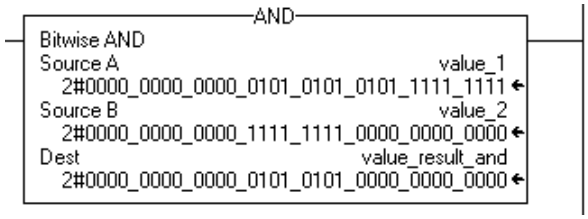
Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Se è abilitata, l'istruzione AND esegue un'operazione AND di bit su SourceA e SourceB e inserisce il risultato nella Dest.



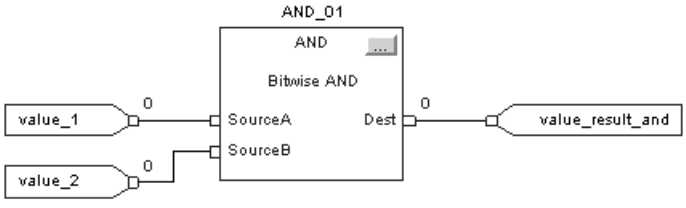
Ladder



Testo strutturato

```
value_result_and := value_1 AND value_2;
```

Blocco funzione

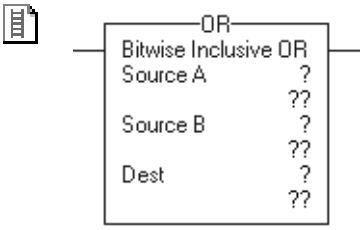


Bitwise OR (OR)

L'istruzione OR esegue un'operazione OR di bit di Source A e Source B e colloca i risultati nella Destinazione.

Per eseguire un OR logico, vedere pagina 317.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore su cui eseguire OR con Source B
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Source B	SINT	immediato	valore su cui eseguire OR con Source A
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Destination	SINT	tag	memorizza il risultato
	INT		
	DINT		

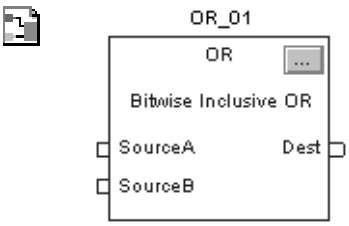
Testo strutturato

```
dest := sourceA OR sourceB
```

Usare OR come operatore all'interno di un'espressione. Questa espressione valuta *sourceA* OR *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
OR tag	FBD_LOGICAL	struttura	struttura OR

Struttura FBD_LOGICAL

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	DINT	Valore su cui eseguire OR con SourceB. Valori validi = qualsiasi numero intero
SourceB	DINT	Valore su cui eseguire OR con SourceA. Valori validi = qualsiasi numero intero
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Risultato dell'istruzione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se è abilitata, l'istruzione valuta l'operazione OR:

Se il bit In Source A è	E il bit in Source B è	Il bit nella Destination è
0	0	0
0	1	1
1	0	1
1	1	1

Se si mischiano tipi di dati interi, l'istruzione riempie con degli 0 i bit superiori dei tipi di dati interi più piccoli in modo che questi abbiano la stessa dimensione dei tipi di dati più grandi.

indicatori di stato aritmetico gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

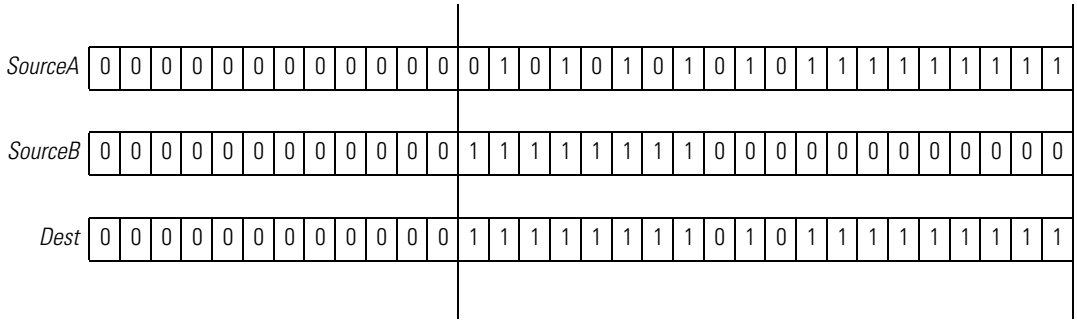
Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione esegue un'operazione OR di bit. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

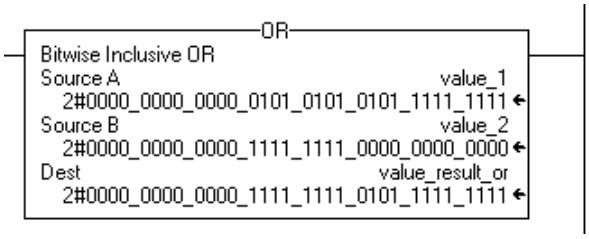
Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Se è abilitata, l'istruzione OR esegue un'operazione OR di bit SourceA e SourceB e inserisce il risultato nella Dest.



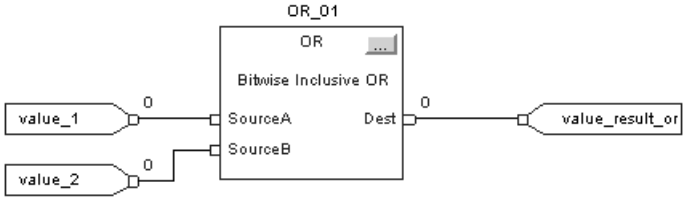
Ladder



Testo strutturato

```
value_result_or := value_1 OR value_2;
```

Blocco funzione

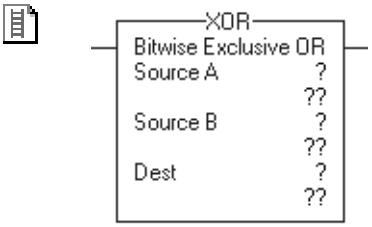


Bitwise Exclusive OR (XOR)

L'istruzione XOR esegue un'operazione XOR di bit utilizzando i bit di Source A e Source B e colloca i risultati nella Destinazione.

Per eseguire un'operazione logica XOR, vedere pagina 320.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source A	SINT	immediato	valore su cui eseguire XOR con Source B
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Source B	SINT	immediato	valore su cui eseguire XOR con Source A
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Destination	SINT	tag	memorizza il risultato
	INT		
	DINT		

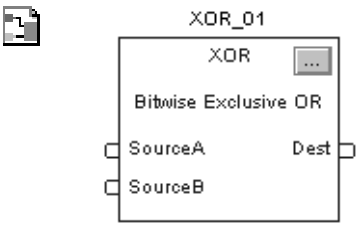
Testo strutturato

```
dest := sourceA XOR sourceB
```

Usare XOR come operatore all'interno di un'espressione. Questa espressione valuta *sourceA* XOR *sourceB*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
XOR tag	FBD_LOGICAL	struttura	struttura XOR

Struttura FBD_LOGICAL

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
SourceA	DINT	Valore su cui eseguire XOR con SourceB. Valori validi = qualsiasi numero intero
SourceB	DINT	Valore su cui eseguire XOR con SourceA. Valori validi = qualsiasi numero intero
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Risultato dell'istruzione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se è abilitata, l'istruzione valuta l'operazione XOR:

Se il bit In Source A è	E il bit in Source B è	Il bit nella Destination è
0	0	0
0	1	1
1	0	1
1	1	0

Se si mischiano tipi di dati interi, l'istruzione riempie con degli 0 i bit superiori dei tipi di dati interi più piccoli in modo che questi abbiano la stessa dimensione dei tipi di dati più grandi.

indicatori di stato aritmetico gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

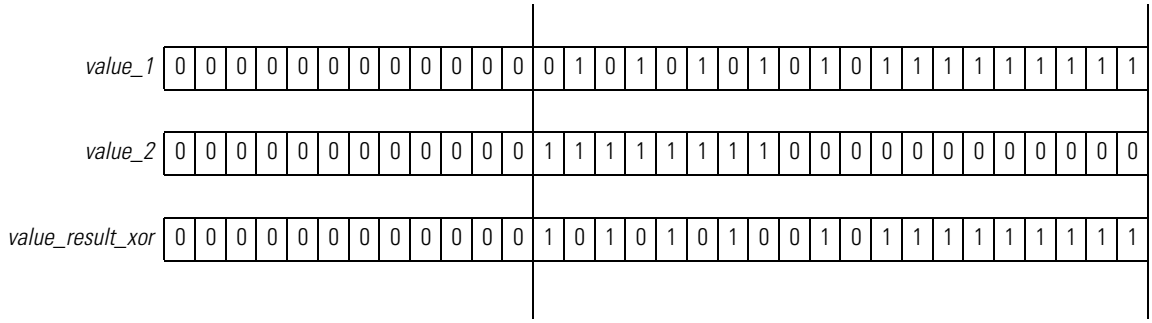
Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione esegue un'operazione OR di bit. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

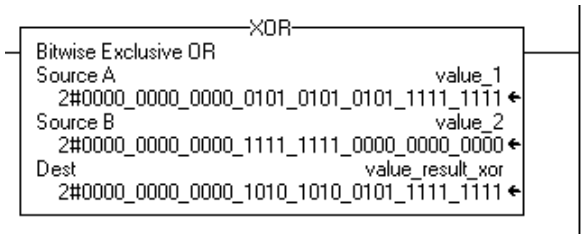
Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Se è abilitata, l'istruzione XOR esegue un'operazione XOR di bit su SourceA e SourceB e inserisce il risultato nel tag di destinazione.



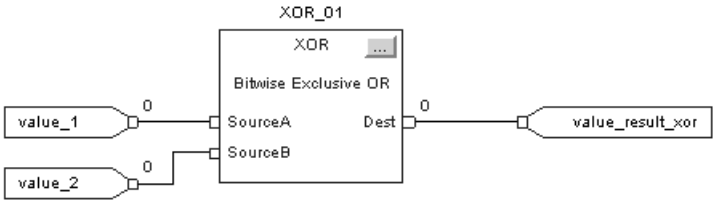
Ladder



Testo strutturato

```
value_result_xor := value_1 XOR value_2;
```

Blocco funzione

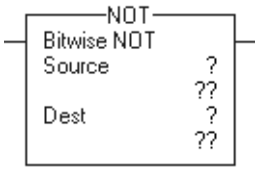


Bitwise NOT (NOT)

L'istruzione NOT esegue un'operazione NOT di bit utilizzando i bit di Source e colloca il risultato nella Destinazione.

Per quanto riguarda l'esecuzione di un'operazione NOT logica, vedere pagina 323.

Operandi:



Ladder

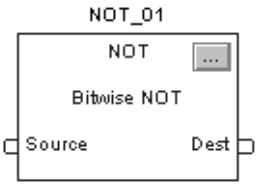
Operando	Tipo	Formato	Descrizione
Source	SINT INT DINT Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.	immediato tag	valore su cui eseguire l'operazione NOT
Destination	SINT INT DINT	tag	memorizza il risultato

```
dest := Source NOT
```

Testo strutturato

Usare NOT come operatore all'interno di un'espressione. Questa espressione valuta NOT *Source*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
NOT tag	FBD_LOGICAL	struttura	struttura NOT

Struttura FBD_LOGICAL

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	DINT	Valore su cui eseguire l'operazione NOT. Valori validi = qualsiasi numero intero
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Risultato dell'istruzione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se è abilitata, l'istruzione valuta l'operazione NOT:

Se il bit nella Source A è	Il bit nella Destinazione sarà
0	1
1	0

Se si mischiano tipi di dati interi, l'istruzione riempie con degli 0 i bit superiori dei tipi di dati interi più piccoli in modo che questi abbiano la stessa dimensione dei tipi di dati più grandi.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	L'istruzione esegue un'operazione NOT di bit. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

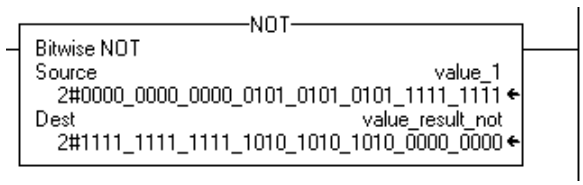
Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Se è abilitata, l'istruzione NOT esegue un'operazione NOT di bit su Source e inserisce il risultato nella Dest.

value_1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1
value_result_not	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0

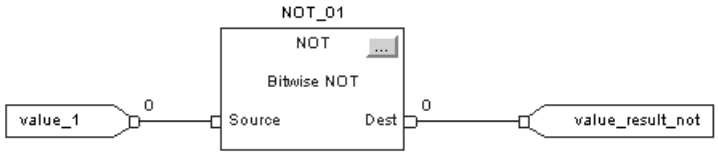
Ladder



Testo strutturato

```
value_result_not := NOT value_1;
```

Blocco funzione



Boolean AND (BAND)

L'istruzione BAND esegue operazioni AND logiche su 8 ingressi booleani.

Per quanto riguarda l'esecuzione di un'operazione AND di bit, vedere pagina 302.

Operandi:

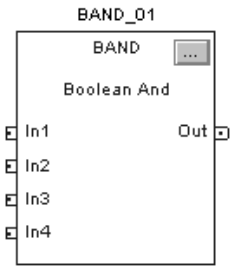
Testo strutturato

```
IF operandA AND operandB THEN
    <istruzione>;
END_IF;
```

Usare AND o il simbolo della e commerciale "&" come operatore all'interno di un'espressione. Gli operandi devono essere valori o espressioni BOOL che valutino i valori BOOL. Questa espressione controlla se *operandA* e *operandB* sono entrambi impostati (true).

Consultare l'Appendice B per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
BAND tag	FBD_BOOLEAN_AND	struttura	struttura BAND

Struttura FBD_BOOLEAN_AND

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
In1	BOOL	Primo ingresso booleano. Il valore di default è impostato.
In2	BOOL	Secondo ingresso booleano. Il valore di default è impostato.
In3	BOOL	Terzo ingresso booleano. Il valore di default è impostato.
In4	BOOL	Quarto ingresso booleano. Il valore di default è impostato.
In5	BOOL	Quinto ingresso booleano. Il valore di default è impostato.
In6	BOOL	Sesto ingresso booleano. Il valore di default è impostato.
In7	BOOL	Settimo ingresso booleano. Il valore di default è impostato.

Parametro di ingresso	Tipo di dati	Descrizione
In8	BOOL	Ottavo ingresso booleano. Il valore di default è impostato.
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	Abilita l'uscita.
Out	BOOL	Uscita dell'istruzione.

Descrizione: L'istruzione BAND esegue operazioni AND logiche su 8 ingressi booleani. In presenza di un ingresso non utilizzato, viene impostato il valore di default (1).

Out = In1 AND In2 AND In3 AND In4 AND In5 AND In6 AND In7 AND In8

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione blocco funzione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

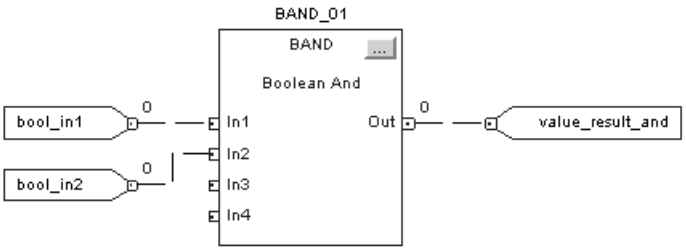
Esempio 1: In questo esempio viene eseguito un AND su *bool_in1* e *bool_in2* e il risultato viene inserito in *value_result_and*.

Se BOOL_IN1 è	Se BOOL_IN2 è	VALUE_RESULT_AND sarà
0	0	0
0	1	0
1	0	0
1	1	1

Testo strutturato

```
value_result_and := bool_in1 AND bool_in2;
```

Blocco funzione



Esempio 2: Se sia *bool_in1* che *bool_in2* sono impostati (true), *light1* viene impostato (on). Diversamente, *light1* viene azzerato (off).

Testo strutturato

```
IF bool_in1 AND bool_in2 THEN
    light1 := 1;
ELSE
    light1 := 0;
END_IF;
```

Boolean OR (BOR)

L'istruzione BOR esegue operazioni OR logiche su 8 ingressi booleani.

Per quanto riguarda l'esecuzione di un'operazione OR di bit, vedere pagina 305.

Operandi:

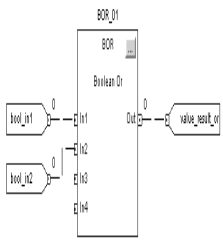
Testo strutturato

```
IF operandA OR operandB THEN
    <istruzione>;
END_IF;
```

Usare OR come operatore all'interno di un'espressione. Gli operandi devono essere valori o espressioni BOOL che valutino i valori BOOL. Questa espressione controlla se *operandA* o *operandB* o entrambi sono impostati (true).

Consultare l'Appendice B per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
BOR tag	FBD_BOOLEAN_OR	struttura	struttura BOR

Struttura FBD_BOOLEAN_OR

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
In1	BOOL	Primo ingresso booleano. Il valore di default è azzerato.
In2	BOOL	Secondo ingresso booleano. Il valore di default è azzerato.
In3	BOOL	Terzo ingresso booleano. Il valore di default è azzerato.
In4	BOOL	Quarto ingresso booleano. Il valore di default è azzerato.
In5	BOOL	Quinto ingresso booleano. Il valore di default è azzerato.
In6	BOOL	Sesto ingresso booleano. Il valore di default è azzerato.
In7	BOOL	Settimo ingresso booleano. Il valore di default è azzerato.

Parametro di ingresso	Tipo di dati	Descrizione
In8	BOOL	Ottavo ingresso booleano. Il valore di default è azzerato.
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	Abilita l'uscita.
Out	BOOL	Uscita dell'istruzione.

Descrizione: L'istruzione BOR esegue operazioni OR logiche su 8 ingressi booleani. In presenza di un ingresso non utilizzato, il valore di default viene azzerato (0).

Out = In1 OR In2 OR In3 OR In4 OR In5 OR In6 OR In7 OR In8

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione blocco funzione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

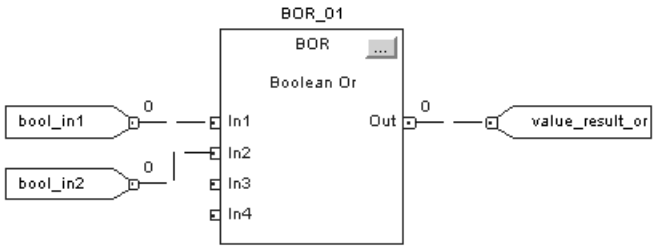
Esempio 1: In questo esempio viene eseguito un OR su *bool_in1* e *bool_in2* e il risultato viene inserito in *value_result_or*.

Se BOOL_IN1 è	Se bool_in2 è	Value_result_or sarà
0	0	0
0	1	1
1	0	1
1	1	1

Testo strutturato

```
value_result_or := bool_in1 OR bool_in2;
```

Blocco funzione



Esempio 2: In questo esempio, *light1* viene impostato (on) se:

- solo *bool_in1* è impostato (true).
- solo *bool_in2* è impostato (true).
- sia *bool_in1* che *bool_in2* sono impostati (true).

Diversamente, *light1* viene azzerato (off).

Testo strutturato

```
IF bool_in1 OR bool_in2 THEN
    light1 := 1;
ELSE
    light1 := 0;
END_IF;
```

Boolean Exclusive OR (BXOR)

BXOR esegue un OR esclusivo su 2 ingressi booleani.

Per quanto riguarda l'esecuzione di un'operazione XOR di bit, vedere pagina 308.

Operandi:

Testo strutturato

```
IF operandA XOR operandB THEN
    <istruzione>;
END_IF;
```

Usare XOR come operatore all'interno di un'espressione. Gli operandi devono essere valori o espressioni BOOL che valutino i valori BOOL. Questa espressione controlla se è impostato (true) solo *operandA* o solo *operandB*.

Consultare l'Appendice B per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione

Operando	Tipo	Formato	Descrizione
BXOR tag	FBD_BOOLEAN_XOR	struttura	struttura BXOR

Struttura FBD_BOOLEAN_XOR

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
In1	BOOL	Primo ingresso booleano. Il valore di default è azzerato.
In2	BOOL	Secondo ingresso booleano. Il valore di default è azzerato.
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	Abilita l'uscita.
Out	BOOL	Uscita dell'istruzione.

Descrizione: L'istruzione BXOR esegue un OR esclusivo su 2 ingressi booleani.

Out = In1 XOR In2

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione blocco funzione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

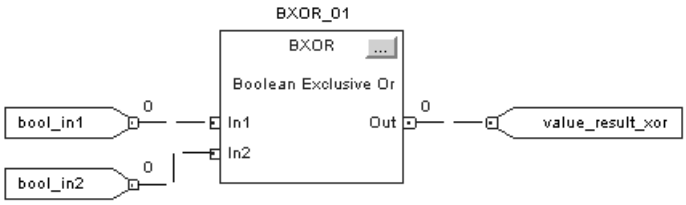
Esempio 1: In questo esempio viene eseguito un OR esclusivo su *bool_in1* e *bool_in2* e il risultato viene inserito in *value_result_xor.*

Se BOOL_IN1 è	Se BOOL_IN2 è	VALUE_RESULT_XOR sarà
0	0	0
0	1	1
1	0	1
1	1	0

Testo strutturato

```
value_result_xor := bool_in1 XOR bool_in2;
```

Blocco funzione



Esempio 2: In questo esempio, *light1* viene impostato (on) se

- solo *bool_in1* è impostato (true).
- solo *bool_in2* è impostato (true).

Diversamente, *light1* viene azzerato (off).

Testo strutturato

```
IF bool_in1 XOR bool_in2 THEN
    light1 := 1;
ELSE
    light1 := 0;
END_IF;
```

Boolean NOT (BNOT)

L'istruzione BNOT complementa un ingresso booleano.

Per quanto riguarda l'esecuzione di un'operazione NOT di bit, vedere pagina 311.

Operandi:

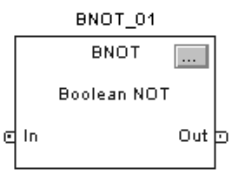
testo strutturato

```
IF NOT operand THEN
    <istruzione>;
END_IF;
```

Usare NOT come operatore all'interno di un'espressione. L'operando deve corrispondere a valori o espressioni BOOL che valutino i valori BOOL. Questa espressione valuta se l'*operando* è azzerato (falso).

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
BNOT tag	FBD_BOOLEAN_NOT	struttura	struttura BNOT

Struttura FBD_BOOLEAN_NOT

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
In	BOOL	Ingresso istruzione. Il valore di default è impostato.
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	Abilita l'uscita.
Out	BOOL	Uscita dell'istruzione.

Descrizione: L'istruzione BNOT complementa un ingresso booleano.

Out = NOT In

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione blocco funzione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

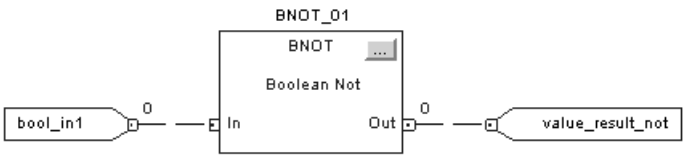
Esempio 1: In questo esempio viene eseguito il complemento di *bool_in1* e il risultato viene inserito in *value_result_not*.

Se BOOL_IN1 è	VALUE_RESULT_NOT sarà
0	1
1	0

Testo strutturato

```
value_result_not := NOT bool_in1;
```

Blocco funzione



Esempio 2: Se *bool_in1* viene azzerato, *light1* viene azzerato (off). Diversamente, *light1* viene impostato (on).

Testo strutturato

```
IF NOT bool_in1 THEN
    light1 := 0;
ELSE
    light1 := 1;
END_IF;
```

Istruzioni di matrice (file)/varie
(FAL, FSC, COP, CPS, FLL, AVE, SRT, STD, SIZE)

Introduzione

Le istruzioni di file/varie operano su matrici di dati.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
eseguire operazioni aritmetiche, logiche, di spostamento e funzionali su valori di matrici	FAL	ladder testo strutturato[1]	331
cercare e confrontare valori di matrici	FSC	ladder	342
copiare il contenuto di una matrice in un'altra matrice	COP	ladder testo strutturato	350
copiare il contenuto di una matrice in un'altra matrice senza interruzione	CPS	ladder testo strutturato	350
riempire una matrice con determinati dati	FLL	ladder testo strutturato[1]	356
calcolare la media di una matrice di valori	AVE	ladder testo strutturato[1]	360
ordinare i dati di una dimensione di una matrice in ordine crescente	SRT	ladder testo strutturato	365
calcolare la deviazione standard di una matrice di valori	STD	ladder testo strutturato[1]	370
trovare la dimensione di una dimensione di una matrice	SIZE	ladder testo strutturato	375

(1) Non esiste un'istruzione equivalente in testo strutturato. Utilizzare un'altra programmazione in testo strutturato per ottenere lo stesso risultato. Vedere la descrizione dell'istruzione.

È possibile usare tipi di dati diversi, ma si potrebbe perdere in precisione ed in arrotondamento ed inoltre l'istruzione potrebbe impiegare più tempo per essere eseguita. Controllare il bit S:V per verificare se il risultato è stato troncato.

Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimali, in genere DINT o REAL.

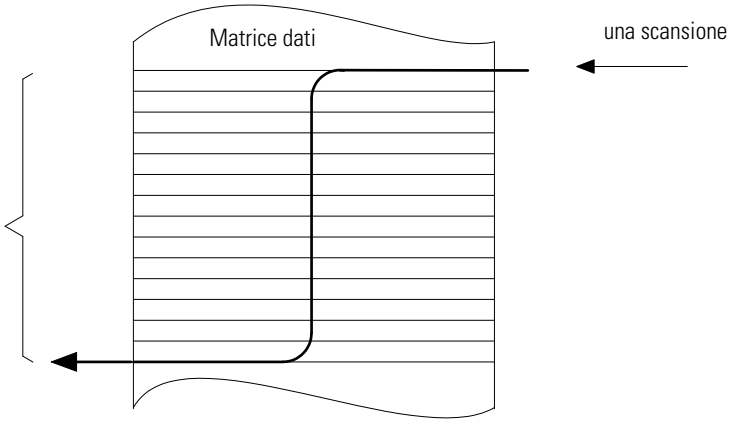
Scelta della modalità operativa

Nel caso delle istruzioni FAL e FSC, la modalità indica al controllore come distribuire le operazioni di matrice.

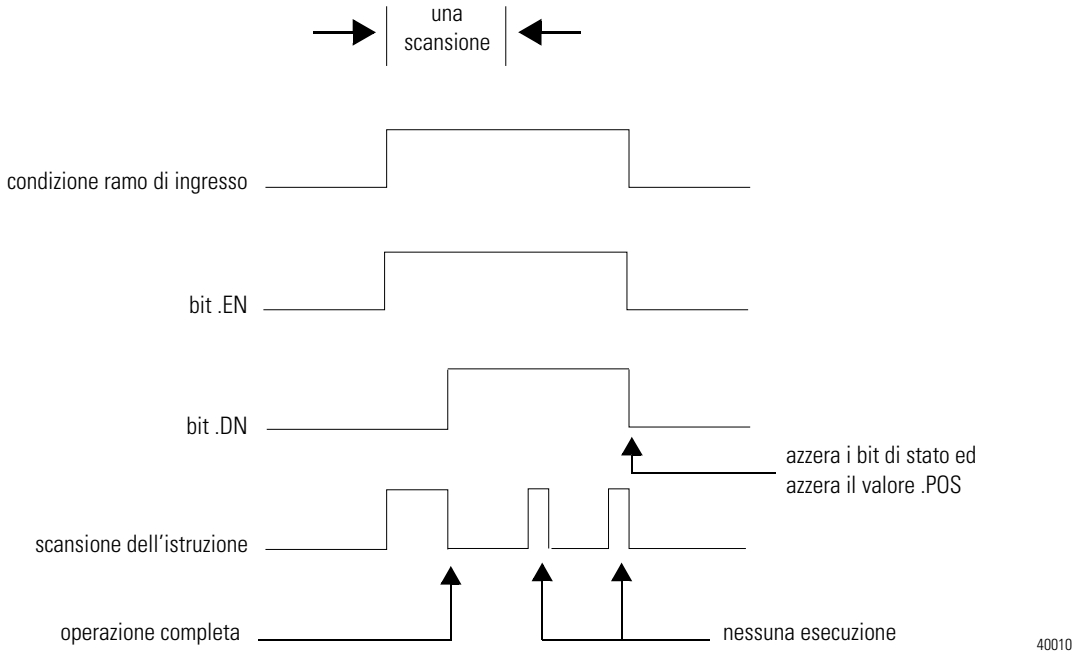
Se si desidera	Selezionare questa modalità
operare su tutti gli elementi specificati in una matrice prima di continuare con l'istruzione successiva	Continua (All)
distribuire l'operazione sulla matrice su più scansioni immettere il numero di elementi per scansione su cui operare (1-2147483647)	Discreta (Numerical)
manipolare un elemento della matrice ogni volta che la condizione del ramo di ingresso passa da falsa a vera	Incrementale (Incremental)

Modalità Continua

In modalità Continua (ALL), si opera su tutti gli elementi specificati nella matrice prima di continuare con l'istruzione successiva. L'operazione inizia quando la condizione del ramo di ingresso dell'istruzione passa da falsa a vera. Il valore posizione (.POS) della struttura di controllo indica l'elemento della matrice che l'istruzione sta utilizzando. L'operazione ha termine quando il valore .POS è uguale al valore .LEN.



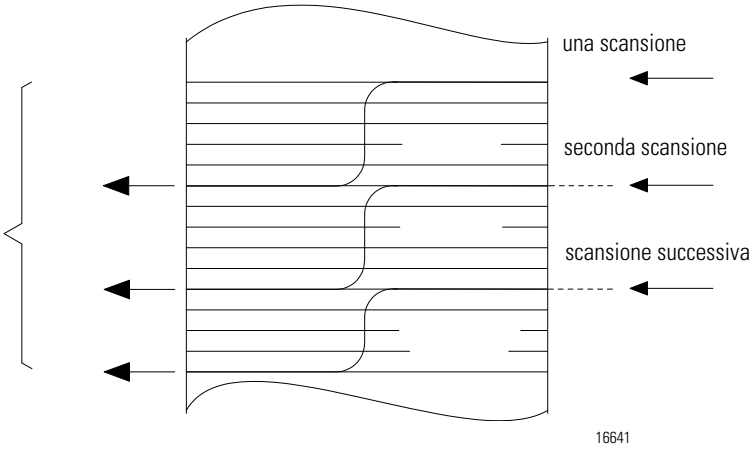
Il seguente schema di temporizzazione mostra la relazione tra i bit di stato e il funzionamento dell'istruzione. Quando l'esecuzione dell'istruzione è completa, viene impostato il bit .DN. Quando la condizione del ramo di ingresso è falsa, il bit .DN, il bit .EN ed il valore .POS vengono azzerati. Solo a questo punto una transizione della condizione del ramo di ingresso da falsa a vera può attivare un'altra esecuzione dell'istruzione.



Modalità discreta

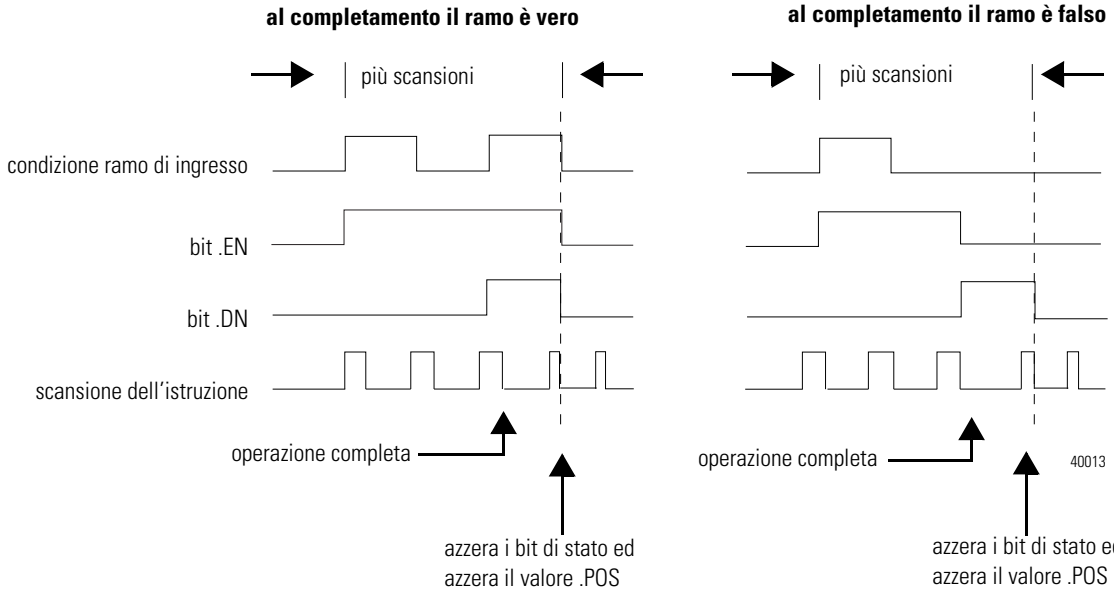
La modalità discreta (NUMERICAL) distribuisce le operazioni sulla matrice in varie scansioni. Questa modalità è utile quando si lavora su dati non critici per tempo o su grandi quantità di dati. Si inserisce il numero di elementi su cui operare ad ogni scansione e così facendo si riducono i tempi di scansione.

L'esecuzione viene attivata quando la condizione del ramo di ingresso passa da falsa a vera. Una volta attivata, l'istruzione viene eseguita ogni volta che viene scandita e per il numero di volte necessario a completare l'operazione su tutta la matrice. Dopo essere stata attivata, la condizione del ramo di ingresso può cambiare continuamente senza interrompere l'esecuzione dell'istruzione.



IMPORTANTE Evitare di utilizzare i risultati di un'istruzione di file funzionante in modalità discreta fino a quando non sia stato impostato il bit .DN.

Il seguente schema di temporizzazione mostra la relazione tra i bit di stato e il funzionamento dell'istruzione. Quando l'esecuzione dell'istruzione è completa, viene impostato il bit .DN.



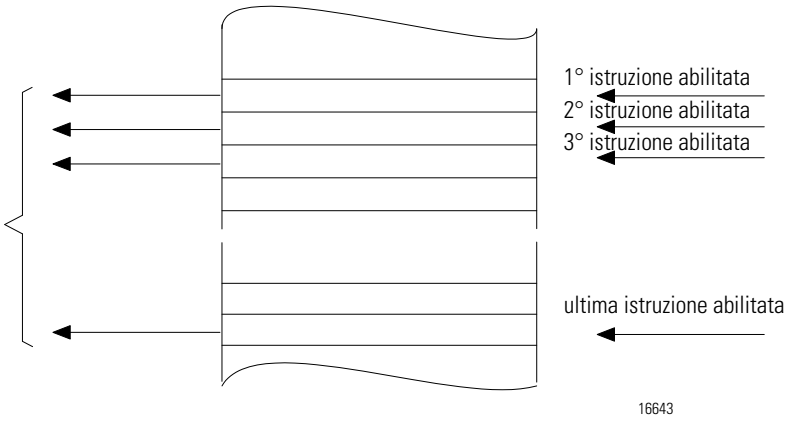
Se al completamento la condizione del ramo di ingresso è vera, i bit .DN ed .ER sono impostati fino a quando la condizione del ramo di

ingresso diventa falsa. Quando la condizione del ramo di ingresso diventa falsa, questi bit vengono azzerati così come il valore .POS.

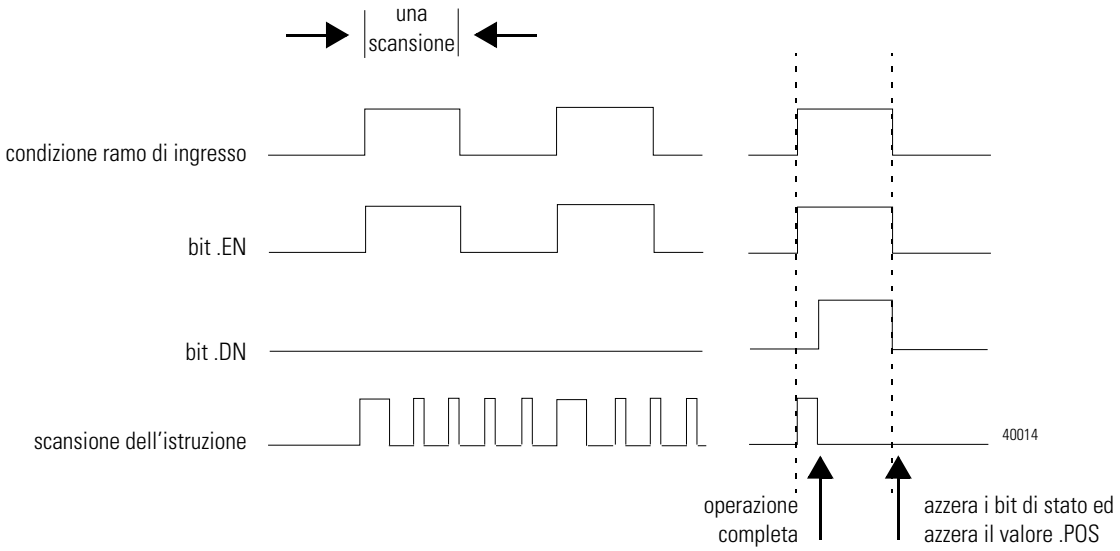
Se al completamento la condizione del ramo di ingresso è falsa, il bit .EN viene azzerato immediatamente. Una scansione dopo l'azzeramento del bit .EN, vengono azzerati il bit .DN ed il valore .POS.

Modalità incrementale

La modalità incrementale (INCREMENTAL) manipola un elemento della matrice ogni volta che la condizione del ramo di ingresso dell'istruzione passa da falsa a vera.



Il seguente schema di temporizzazione mostra la relazione tra i bit di stato e il funzionamento dell'istruzione. L'esecuzione avviene solo in una scansione in cui la condizione del ramo di ingresso passa da falsa a vera. Ogni volta che ciò si verifica viene manipolato un solo elemento della matrice. Se la condizione del ramo di ingresso rimane vera per più di una scansione, l'istruzione viene eseguita solamente durante la prima scansione.



Quando la condizione del ramo di ingresso è vera, viene impostato il bit .EN. Quando è stato manipolato l'ultimo elemento della matrice, viene impostato il bit .DN. Quando l'ultimo elemento è stato manipolato e la condizione del ramo di ingresso diventa falsa, il bit .EN, il bit .DN ed il valore .POS vengono azzerati.

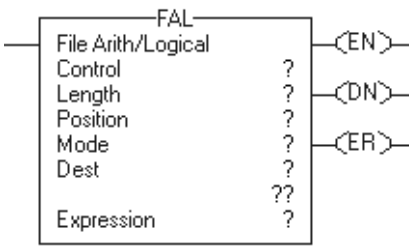
La differenza tra la modalità incrementale e la modalità discreta alla frequenza di un elemento per scansione è che:

- per iniziare l'esecuzione, la modalità discreta con un qualsiasi numero di elementi per scansione richiede solo una transizione da falsa a vera della condizione del ramo di ingresso. L'istruzione continua l'esecuzione del numero di elementi specificato per ogni scansione fino al suo completamento, a prescindere dallo stato della condizione del ramo di ingresso.
- La modalità incrementale richiede un cambiamento della condizione del ramo di ingresso da falsa a vera affinché sia possibile manipolare un elemento della matrice.

File Arithmetic and Logic (FAL)

L'istruzione FAL esegue operazioni di copia, aritmetiche, logiche e di funzione sui dati memorizzati in una matrice.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Control	CONTROL	tag	struttura di controllo per l'operazione
Length	DINT	immediato	numero di elementi della matrice da manipolare
Position	DINT	immediato	elemento corrente della matrice il valore iniziale generalmente è 0
Mode	DINT	immediato	come distribuire l'operazione selezionare INC, ALL oppure inserire un numero
Destination	SINT INT **DINT** **REAL**	tag	tag dove memorizzare il risultato
Expression	SINT INT **DINT** **REAL**	immediato tag	un'espressione formata da tag e/o valori immediati separati da operatori
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		

Testo strutturato

Il linguaggio testo strutturato non prevede un'istruzione FAL, tuttavia è possibile ottenere lo stesso risultato utilizzando un'istruzione SIZE e un FOR...DO o altro costrutto loop.

```
SIZE(destination,0,length-1);
FOR position = 0 TO length DO
    destination[position] := numeric_expression;
END_FOR;
```

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi dei costrutti in testo strutturato.

Struttura di controllo

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione FAL è abilitata.
.DN	BOOL	Il bit di fine viene impostato quando l'istruzione ha operato sull'ultimo elemento (.POS = .LEN).
.ER	BOOL	Il bit di errore viene impostato se l'espressione genera un overflow (S:V è impostato). L'istruzione interrompe la sua esecuzione fino a quando il programma non azzera il bit .ER. Il valore .POS contiene la posizione dell'elemento che ha causato un overflow.
.LEN	DINT	La lunghezza specifica il numero di elementi della matrice su cui opera l'istruzione FAL.
.POS	DINT	La posizione contiene la posizione dell'elemento corrente a cui l'istruzione accede.

Descrizione: L'istruzione FAL esegue sulle matrici le stesse operazioni che l'istruzione CPT esegue sugli elementi.

Gli esempi che iniziano a pagina 338 mostrano come usare il valore POS per passare attraverso una matrice. Se un indice dell'espressione della destinazione è fuori gamma, l'istruzione FAL genera un errore grave (tipo 4, codice 20).

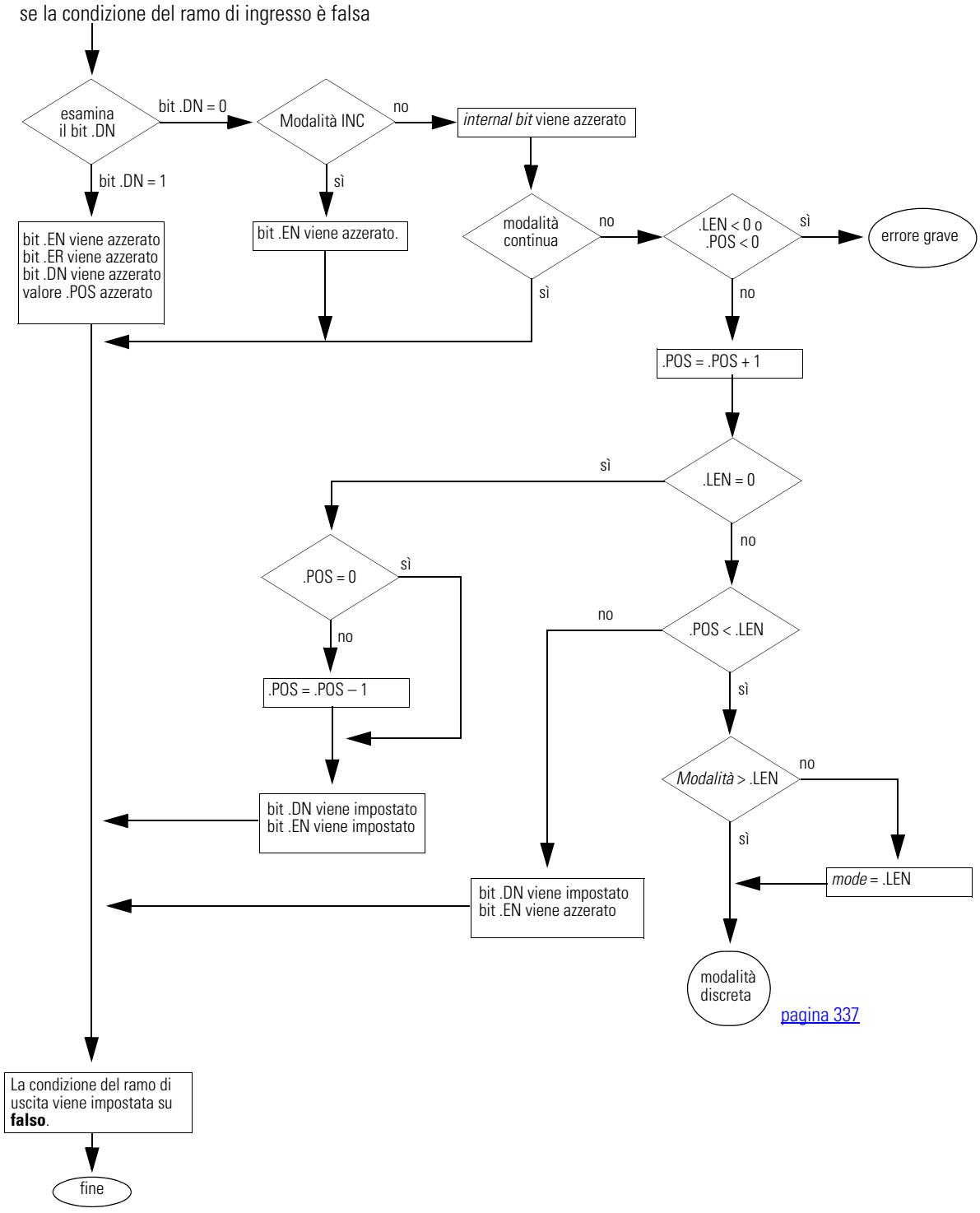
Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

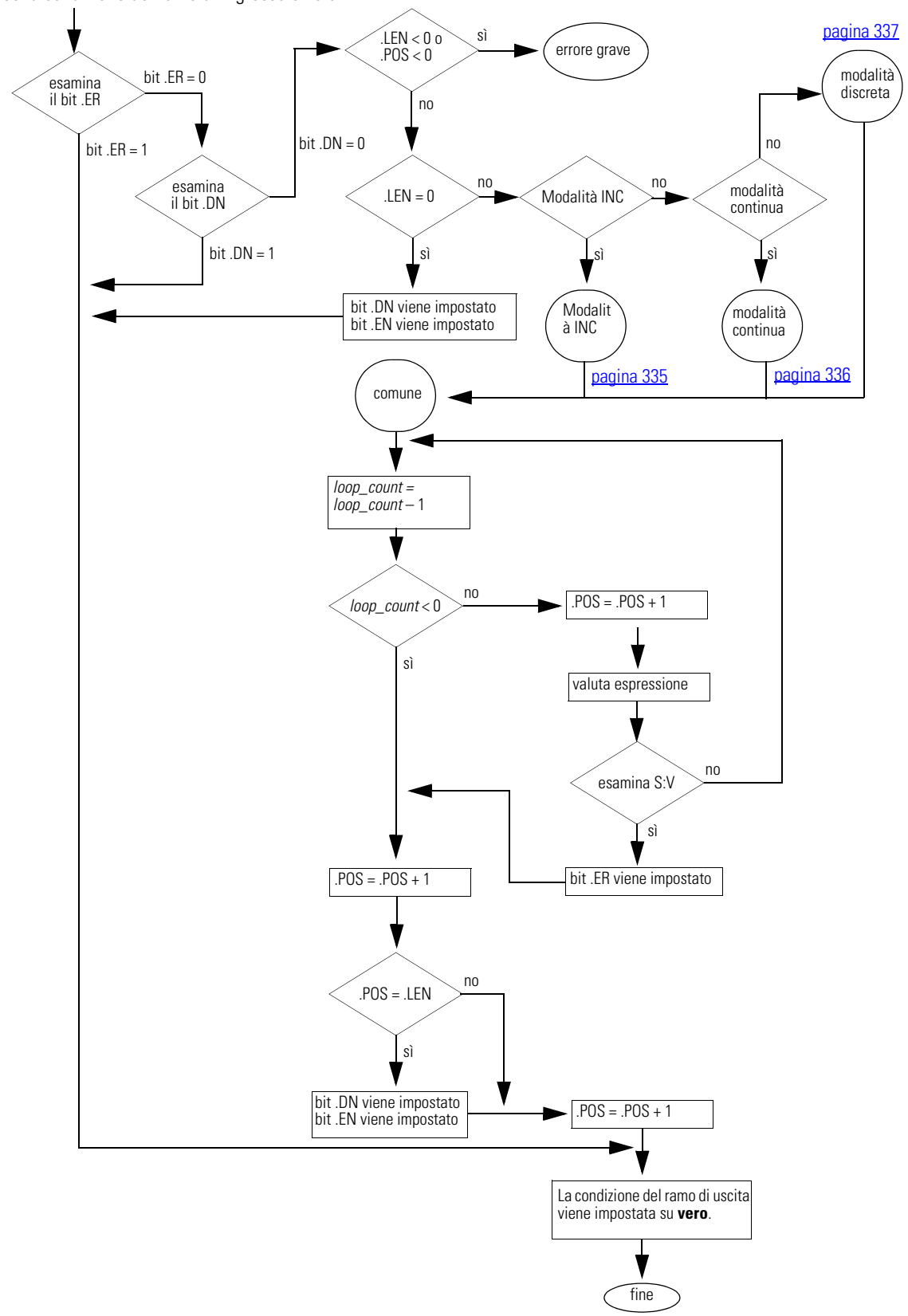
Si verifica un errore grave se	Tipo errore	Codice errore
l'indice è fuori gamma	4	20
.POS < 0 oppure .LEN < 0	4	21

Esecuzione:

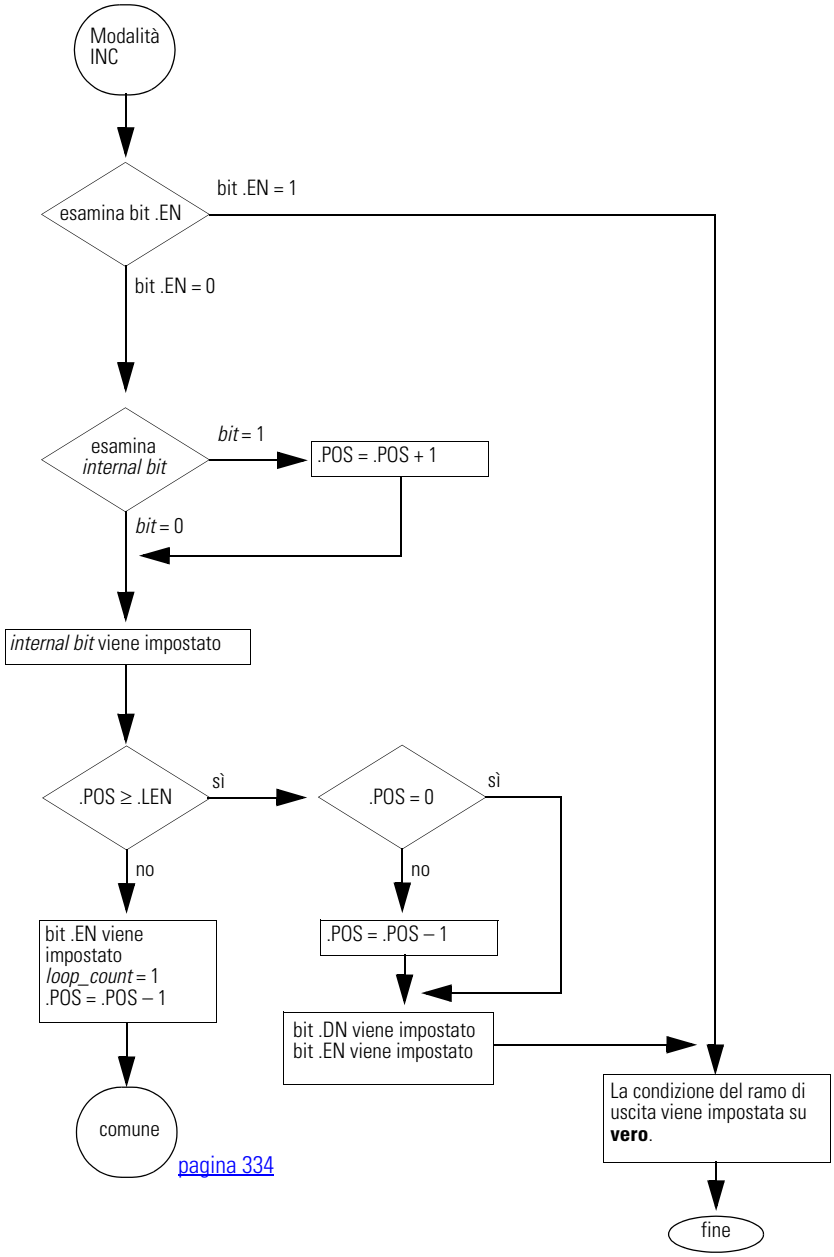
Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.



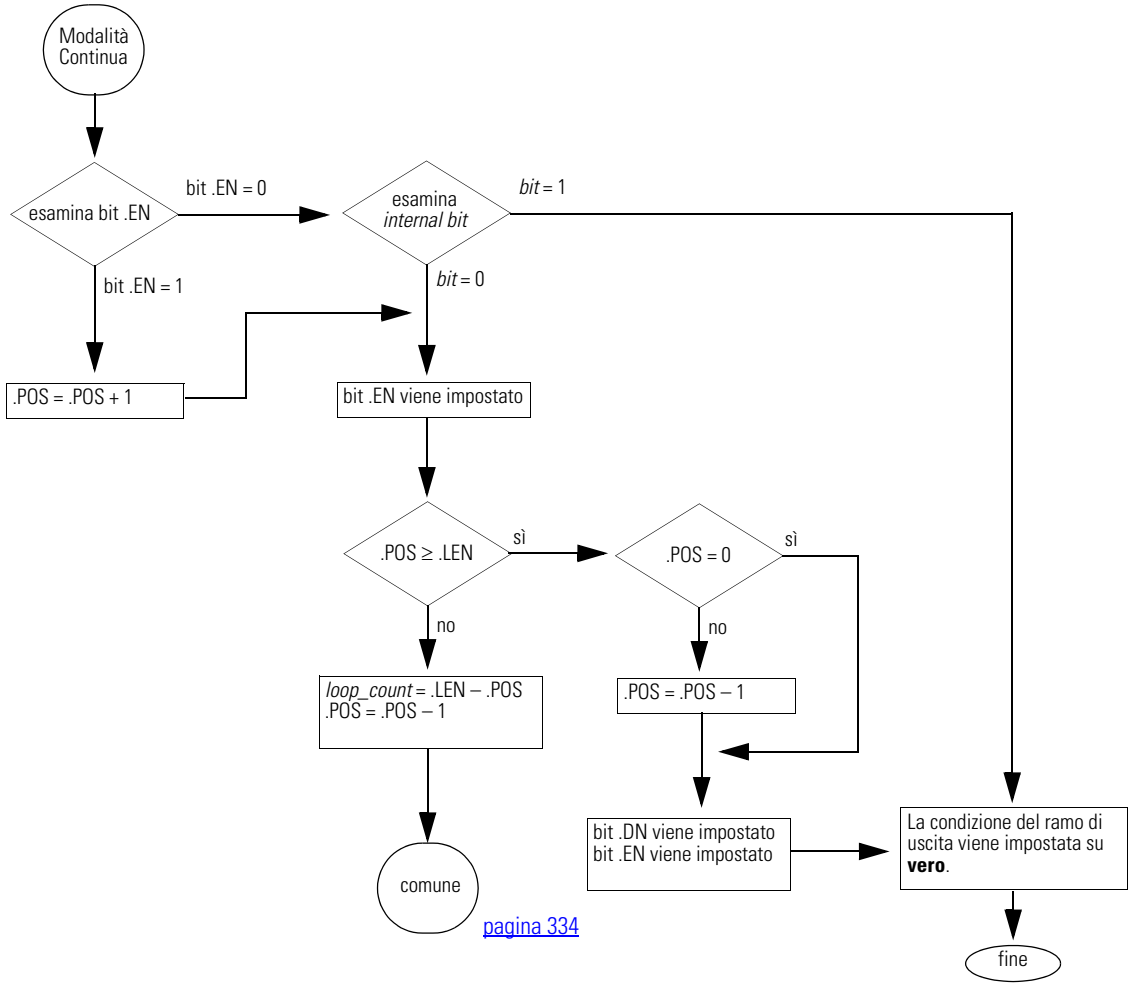
Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	



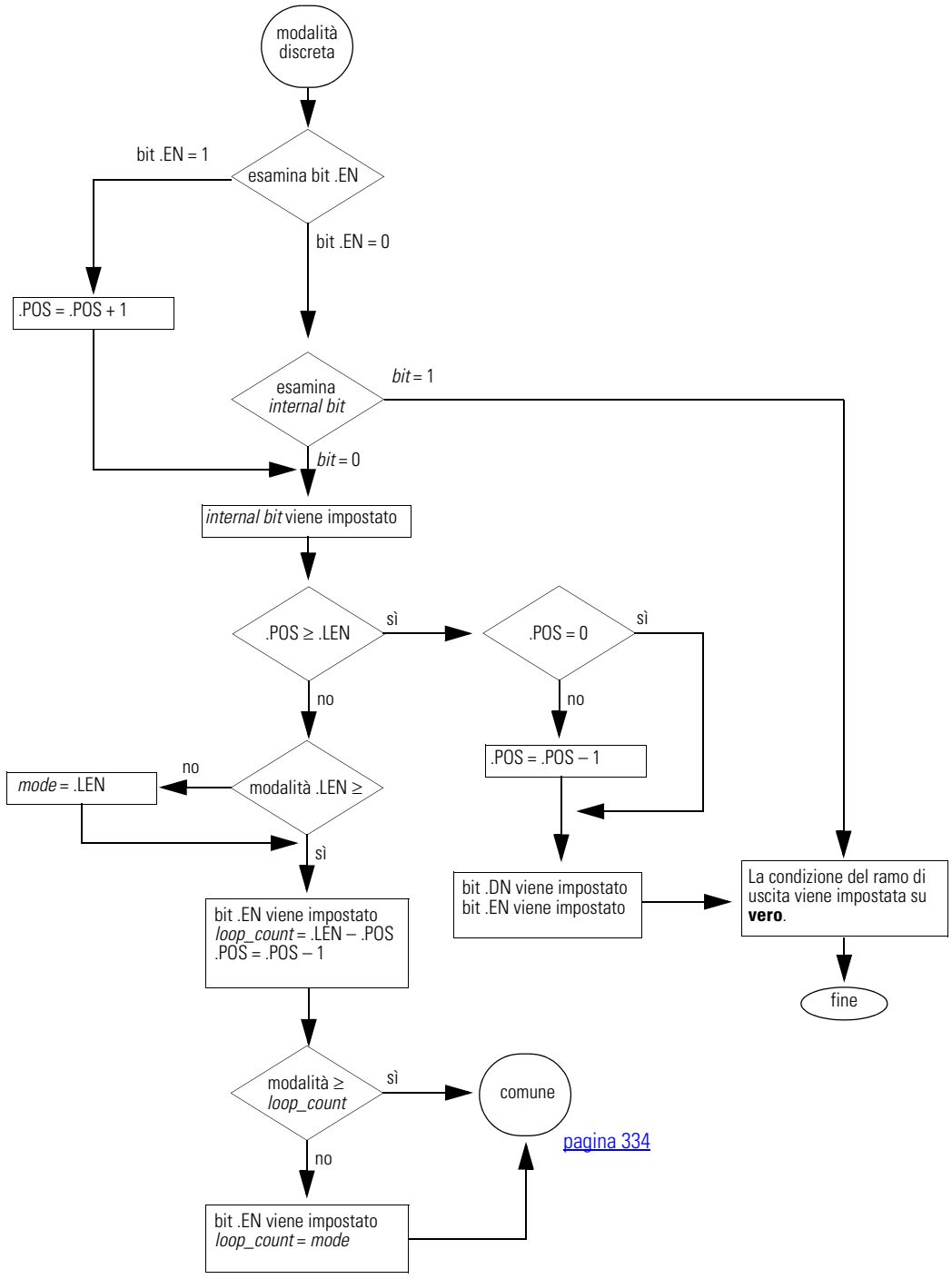
Condizione	Azione ladder



Condizione	Azione ladder

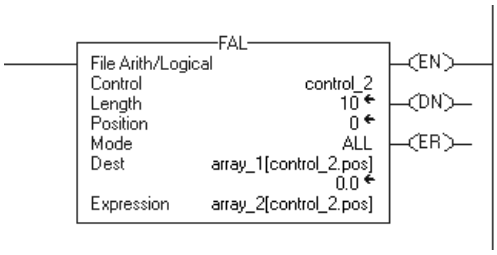


Condizione	Azione ladder

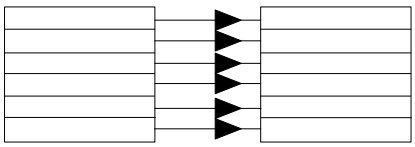


post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio 1: Se è abilitata, l'istruzione FAL copia ciascun elemento di *array_2* nella stessa posizione di *array_1*.



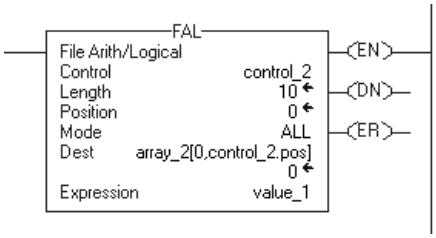
copia da matrice a matrice



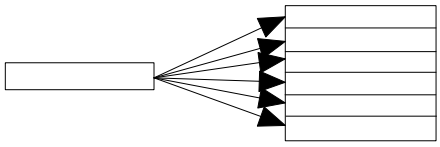
Espressione: *array_2[control_2.pos]*

Destination: *array_1[control_2.pos]*

Esempio 2: Se è abilitata, l'istruzione FAL copia *value_1* nelle prime 10 posizioni della seconda dimensione di *array_2*.



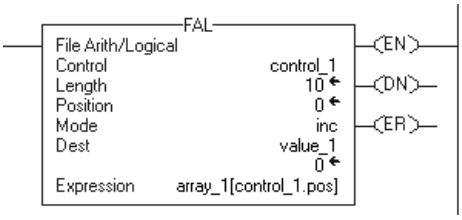
copia da elemento a matrice



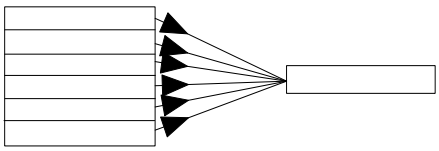
Espressione: *value_1*

Destination: *array_2[0,control_2.pos]*

Esempio 3: Ogni volta che l'istruzione FAL è abilitata, questa copia il valore corrente di *array_1* in *value_1*. L'istruzione FAL utilizza la modalità incrementale, pertanto, ogni volta che l'istruzione viene abilitata, viene copiato solo un elemento alla volta. La volta successiva che viene abilitata, l'istruzione sovrascrive *value_1* con il valore successivo di *array_1*.



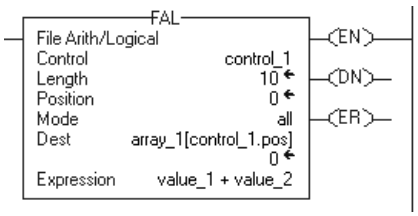
copia da matrice ad elemento



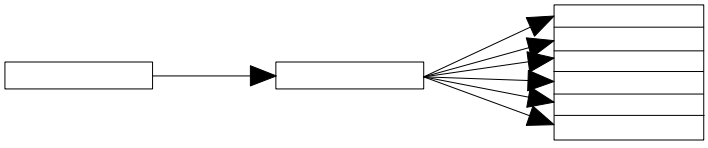
Espressione: *array_1[control_1.pos]*

Destination: *value_1*

Esempio 4: Se è abilitata, l'istruzione FAL somma *value_1* e *value_2* e memorizza il risultato nella posizione corrente di *array_1*.



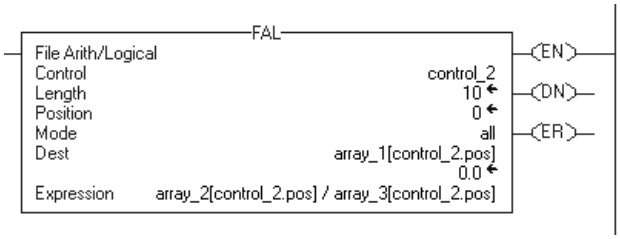
operazione aritmetica: (elemento + elemento) a matrice



Espressione: *value_1 + value_2*

Destination: *array_1[control_1.pos]*

Esempio 5: Se è abilitata, l'istruzione FAL divide il valore della posizione corrente di *array_2* con il valore della posizione corrente di *array_3* e memorizza il risultato nella posizione corrente di *array_1*.

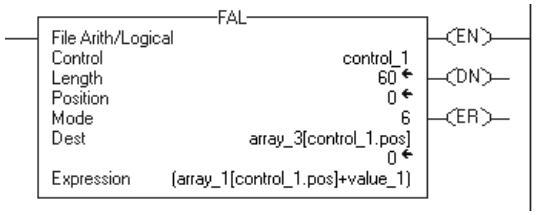


operazione aritmetica: (matrice/matrice) a matrice

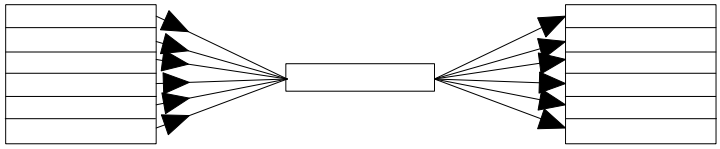
Espressione:
array_2[control_2.pos]/array_3[control_2.pos]

Destination:
array_1[control_2.pos]

Esempio 6: Se è abilitata, l'istruzione FAL somma il valore della posizione corrente di *array_1* e di *value_1* e memorizza il risultato nella posizione corrente di *array_3*. L'istruzione deve essere eseguita 10 volte affinché gli interi *array_1* e *array_3* vengano elaborati.



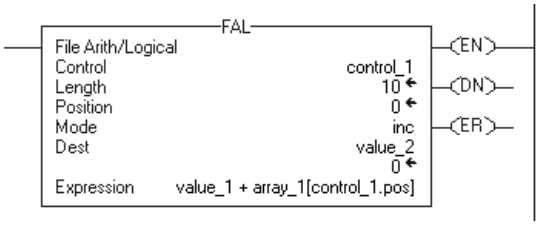
operazione aritmetica: (matrice + elemento) a matrice



Espressione:
array_1[control_1.pos] + value_1

Destination:
array_3[control_1.pos]

Esempio 7: Ogni volta che l'istruzione FAL è abilitata, essa somma *value_1* ed il valore corrente di *array_1* e memorizza il risultato in *value_2*. L'istruzione FAL utilizza la modalità incrementale, pertanto, ogni volta che l'istruzione viene abilitata, solo un elemento viene sommato a *value_1*. La volta successiva che l'istruzione verrà abilitata, l'istruzione sovrascriverà il *value_2*.

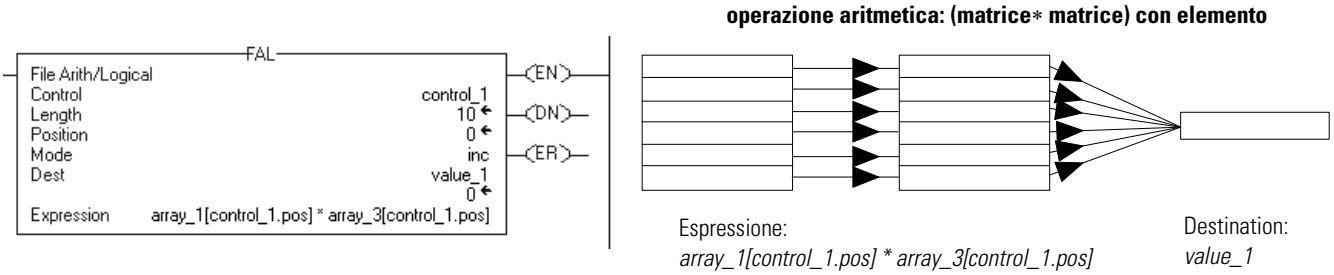


operazione aritmetica: (elemento + matrice) ad elemento

Espressione:
value_1 + array_1[control_1.pos]

Destination:
value_2

Esempio 8: Se è abilitata, l'istruzione FAL moltiplica il valore corrente di *array_1* per il valore corrente di *array_3* e memorizza il risultato in *value_1*. L'istruzione FAL utilizza la modalità incrementale, pertanto, ogni volta che l'istruzione viene abilitata, viene moltiplicata solo una coppia di elementi. La volta successiva che verrà abilitata, l'istruzione sovrascriverà *value_1*.



Espressioni FAL

Le espressioni delle istruzioni FAL vanno programmate nello stesso modo delle espressioni delle istruzioni CPT. Per informazioni su operatori validi, formato ed ordine delle operazioni comuni ad entrambi le istruzioni, utilizzare le seguenti sezioni.

Operatori validi

Operatore	Descrizione	Ottimale
+	somma	DINT, REAL
–	sottrai/nega	DINT, REAL
*	moltiplica	DINT, REAL
/	dividi	DINT, REAL
**	esponente (x alla y)	DINT, REAL
ABS	valore assoluto	DINT, REAL
ACS	arcocoseno	REAL
AND	AND di bit	DINT
ASN	arcoseno	REAL
ATN	arcotangente	REAL
COS	coseno	REAL
DEG	da radianti a gradi	DINT, REAL
FRD	da BCD a intero	DINT
LN	logaritmo naturale	REAL

Operatore	Descrizione	Ottimale
LOG	logaritmo in base 10	REAL
MOD	modulo-divide	DINT, REAL
NOT	complemento di bit	DINT
OR	OR di bit	DINT
RAD	da gradi a radianti	DINT, REAL
SIN	seno	REAL
SQR	radice quadrata	DINT, REAL
TAN	tangente	REAL
TOD	da intero a BCD	DINT
TRN	tronca	DINT, REAL
XOR	OR di bit esclusivo	DINT

Formattazione delle espressioni

Per ogni operatore utilizzato in un'espressione, è necessario fornire uno o due operandi (tag o valori immediati). Utilizzare la seguente tabella per formare operatori ed operandi in un'espressione:

Per operatori con:	Utilizzare questo Formato	Esempi:
un operando	operatore(operando)	ABS(*tag_a*)
due operandi	operando_a operatore operando_b	• *tag_b* + 5 • *tag_c* AND *tag_d* • (*tag_e* ** 2) MOD (*tag_f*/*tag_g*)

Determinazione dell'ordine delle operazioni

Le operazioni che vengono scritte nell'espressione sono eseguite dall'istruzione in un determinato ordine, non necessariamente nell'ordine con cui sono state scritte. È possibile cambiare l'ordine delle operazioni raggruppando i termini all'interno di parentesi, forzando l'istruzione a eseguire un'operazione all'interno delle parentesi prima di altre operazioni.

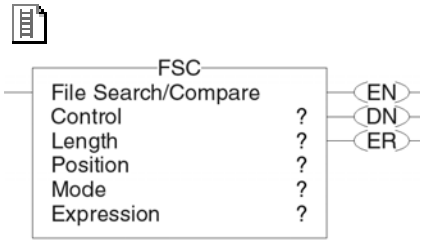
Operazioni di ordine uguale vengono eseguite da sinistra a destra.

Ordine	Funzionamento
1.	( )
2.	ABS, ACS, ASN, ATN, COS, DEG, FRD, LN, LOG, RAD, SIN, SQR, TAN, TOD, TRN
3.	**
4.	– (nega), NOT
5.	*, /, MOD
6.	– (sottrai), +
7.	AND
8.	XOR
9.	OR

File Search and Compare (FSC)

L'istruzione FSC confronta i valori di una matrice, elemento per elemento.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Control	CONTROL	tag	struttura di controllo per l'operazione
Length	DINT	immediato	numero di elementi della matrice da manipolare
Position	DINT	immediato	offset nella matrice il valore iniziale generalmente è 0

Struttura di CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione FSC è abilitata.
.DN	BOOL	Il bit di fine viene impostato quando l'istruzione ha operato sull'ultimo elemento (.POS = .LEN).
.ER	BOOL	Il bit di errore non è modificato.
.IN	BOOL	Il bit di inibizione indica che l'istruzione FSC ha rilevato un confronto vero. Per continuare l'operazione, bisogna azzerare questo bit.
.FD	BOOL	Il bit di trovato indica che l'istruzione FSC ha rilevato un confronto vero.
.LEN	DINT	La lunghezza specifica il numero di elementi della matrice su cui opera l'istruzione.
.POS	DINT	La posizione contiene la posizione dell'elemento corrente a cui l'istruzione accede.

Descrizione:

Quando l'istruzione FSC è abilitata ed il confronto risulta vero, l'istruzione imposta il bit .FD ed il bit .POS indica la posizione della matrice in cui l'istruzione ha trovato il confronto vero. L'istruzione imposta il bit .IN per interrompere la ricerca.

Indicatori di stato aritmetico:

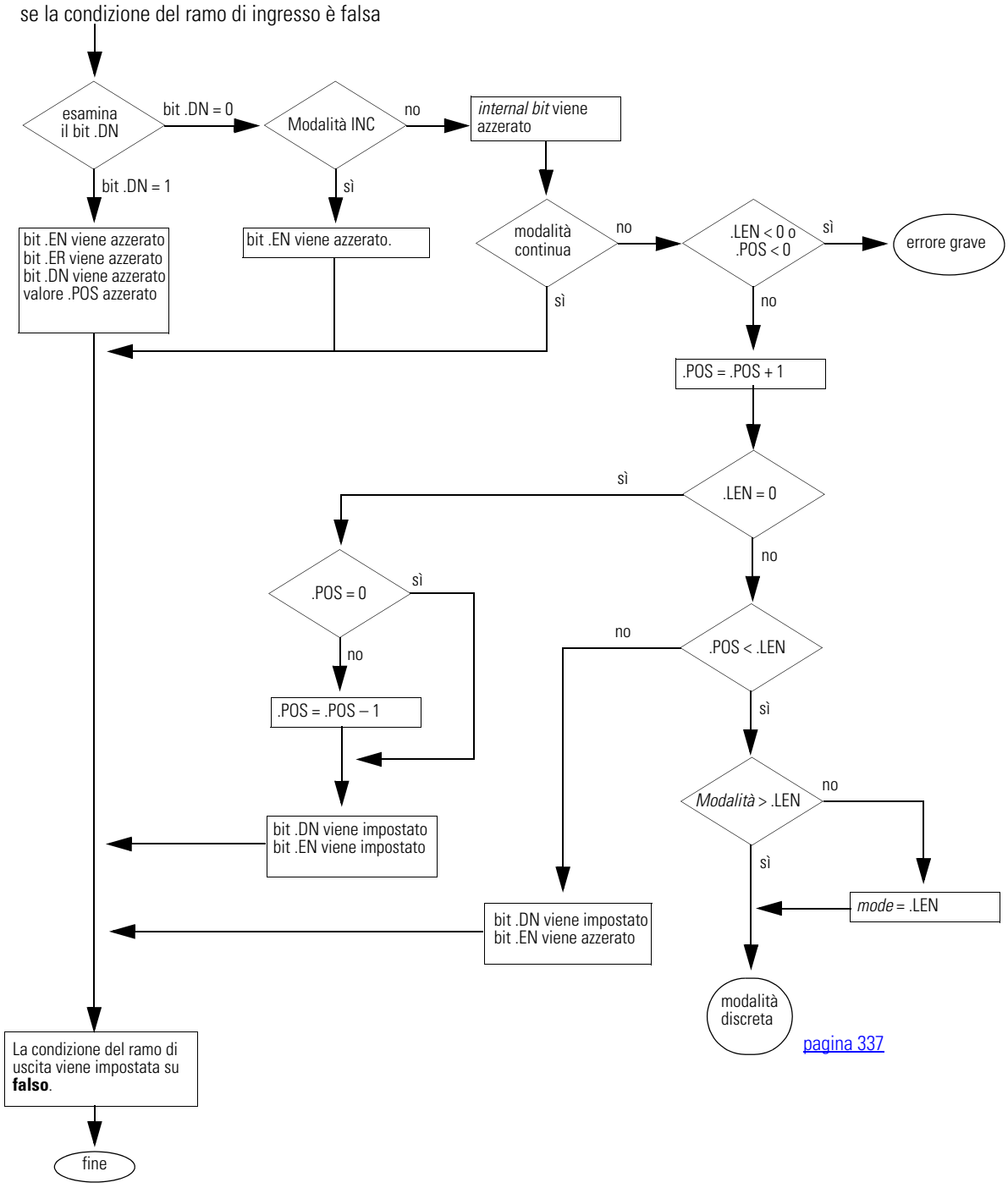
gli indicatori di stato aritmetici sono influenzati.

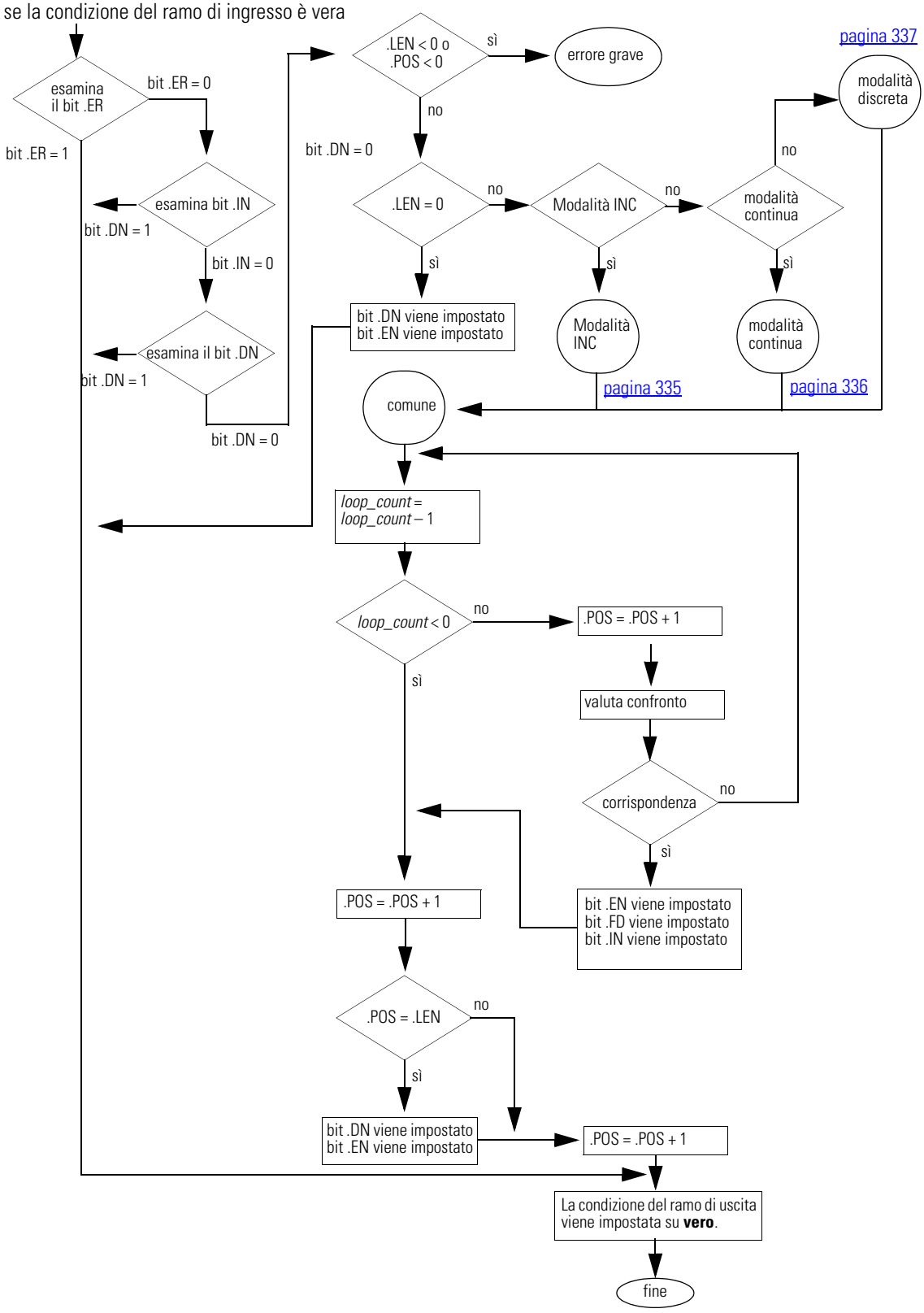
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
.POS < 0 oppure .LEN < 0	4	21

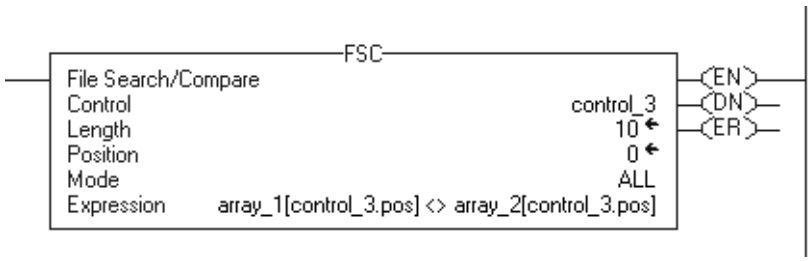
Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.



Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

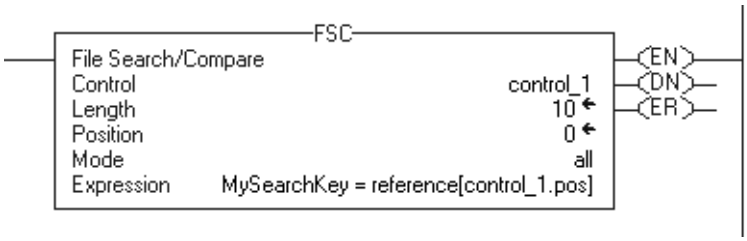
Esempio 1: Ricerca di una corrispondenza tra due matrici. Quando è abilitata l'istruzione FSC confronta ognuno dei primi 10 elementi di *array_1* con i corrispondenti elementi di *array_2*.



array_1	*array_2*	*control_3.pos*
00000000000000000000000000000000	00000000000000000000000000000000	0
00000000000000000000000000000000	00000000000000000000000000000000	1
00000000000000000000000000000000	00000000000000000000000000000000	2
00000000000000000000000000000000	00000000000000000000000000000000	3
00000000000000001111111111111111	11111111111111110000000000000000	4 →
11111111111111111111111111111111	11111111111111111111111111111111	5
11111111111111111111111111111111	11111111111111111111111111111111	6
11111111111111111111111111111111	11111111111111111111111111111111	7
11111111111111111111111111111111	11111111111111111111111111111111	8
11111111111111111111111111111111	11111111111111111111111111111111	9

L'istruzione FSC rivela che questi elementi sono diversi. L'istruzione imposta i bit .FD e .IN. Il valore .POS (4) indica la posizione degli elementi che sono diversi. Per continuare a confrontare il resto della matrice, bisogna azzerare il bit .IN.

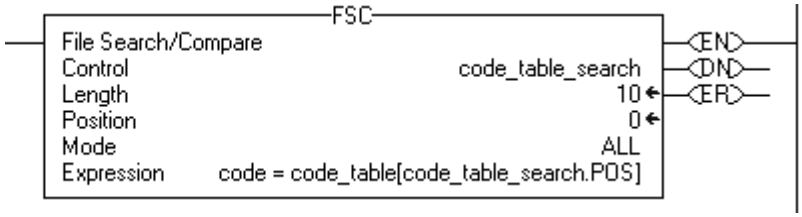
Esempio 2: Ricerca di una corrispondenza in una matrice. Quando è abilitata, l'istruzione FSC confronta *MySearchKey* con i 10 elementi di *array_1*.



MySearchKey	*reference*	*control_3.pos*
	00000000000000000000000000000000	0
	00000000000000000000000000000000	1
	00000000000000000000000000000000	2
	00000000000000000000000000000000	3
11111111111111110000000000000000	11111111111111110000000000000000	4 →
	11111111111111111111111111111111	5
	11111111111111111111111111111111	6
	11111111111111111111111111111111	7
	11111111111111111111111111111111	8
	11111111111111111111111111111111	9

L'istruzione FSC rivela che questo elemento di matrice è uguale a *MySearchKey*. L'istruzione imposta i bit .FD e .IN. Il valore .POS (4) indica la posizione degli elementi uguali. Per continuare a confrontare il resto della matrice, bisogna azzerare il bit .IN.

Esempio 3: Ricerca di una stringa in una matrice di stringhe. Quando è abilitata, l'istruzione FSC confronta i caratteri di *code* a 10 elementi di *code_table*.



code	*code_table*	*code_table_search.POS*
	AFG	0
	BEH	1
	HUO	2
	SAK	3
SAM	SAM	4
	FQG	5
	CLE	6
	CAK	7
	DET	8
	BWG	9

L'istruzione FSC rivela che questo elemento di matrice è uguale a *code*. L'istruzione imposta i bit .FD e .IN. Il valore .POS (4) indica la posizione degli elementi uguali. Per continuare a confrontare il resto della matrice, bisogna azzerare il bit .IN.

Espressioni FSC

Le espressioni delle istruzioni FSC vanno programmate nello stesso modo delle espressioni delle istruzioni CMP. Per informazioni su operatori validi, formato ed ordine delle operazioni comuni ad entrambi le istruzioni, utilizzare le seguenti sezioni.

Operatori validi

Operatore	Descrizione	Ottimale
+	somma	DINT, REAL
–	sottrai/nega	DINT, REAL
*	moltiplica	DINT, REAL
/	dividi	DINT, REAL
=	uguale	DINT, REAL
<	minore di	DINT, REAL
<=	minore di o uguale a	DINT, REAL
>	maggiore di	DINT, REAL
>=	maggiore di o uguale a	DINT, REAL
<>	diverso	DINT, REAL
**	esponente (x alla y)	DINT, REAL
ABS	valore assoluto	DINT, REAL
ACS	arcocoseno	REAL
AND	AND di bit	DINT
ASN	arcoseno	REAL
ATN	arcotangente	REAL
COS	coseno	REAL

Operatore	Descrizione	Ottimale
DEG	da radianti a gradi	DINT, REAL
FRD	da BCD a intero	DINT
LN	logaritmo naturale	REAL
LOG	logaritmo in base 10	REAL
MOD	modulo-divide	DINT, REAL
NOT	complemento di bit	DINT
OR	OR di bit	DINT
RAD	da gradi a radianti	DINT, REAL
SIN	seno	REAL
SQR	radice quadrata	DINT, REAL
TAN	tangente	REAL
TOD	da intero a BCD	DINT
TRN	tronca	DINT, REAL
XOR	OR di bit esclusivo	DINT

Formattazione delle espressioni

Per ogni operatore utilizzato in un'espressione, è necessario fornire uno o due operandi (tag o valori immediati). Utilizzare la seguente tabella per formare operatori ed operandi in un'espressione:

Per operatori con:	Utilizzare questo Formato	Esempi:
un operando	operatore(operando)	ABS(*tag_a*)
due operandi	operando_a operatore operando_b	• *tag_b* + 5 • *tag_c* AND *tag_d* • (*tag_e* ** 2) MOD (*tag_f*/*tag_g*)

Determinazione dell'ordine delle operazioni

Le operazioni che vengono scritte nell'espressione sono eseguite dall'istruzione in un determinato ordine, non necessariamente nell'ordine con cui sono state scritte. È possibile cambiare l'ordine delle operazioni raggruppando i termini all'interno di parentesi, forzando l'istruzione a eseguire un'operazione all'interno delle parentesi prima di altre operazioni.

Operazioni di ordine uguale vengono eseguite da sinistra a destra.

Ordine	Funzionamento
1.	( )
2.	ABS, ACS, ASN, ATN, COS, DEG, FRD, LN, LOG, RAD, SIN, SQR, TAN, TOD, TRN
3.	**
4.	– (nega), NOT
5.	*, /, MOD
6.	<, <=, >, >=, =
7.	– (sottrai), +
8.	AND
9.	XOR
10.	OR

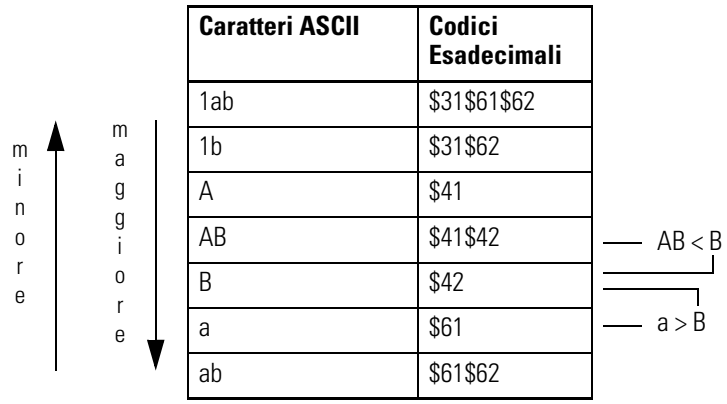
Uso di stringhe in un'espressione

Per utilizzare stringhe di caratteri ASCII in un'espressione, attenersi alle seguenti regole:

- Un'espressione consente di confrontare due tag stringa.
- *Non* è possibile inserire caratteri ASCII direttamente nell'espressione.
- Sono permessi solamente i seguenti operatori

Operatore	Descrizione
=	uguale
<	minore di
<=	minore di o uguale a
>	maggiore di
>=	maggiore di o uguale a
<>	diverso

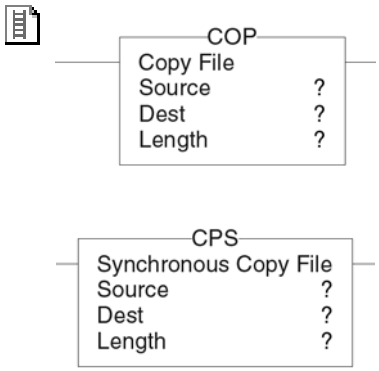
- Le stringhe sono uguali se i loro caratteri corrispondono.
- I caratteri ASCII sono sensibili alle maiuscole. "A" maiuscola ($41) *non* è uguale ad "a" minuscola ($61).
- I valori esadecimali dei caratteri determinano se una stringa è minore o maggiore di un'altra stringa. Per il codice esadecimale di un carattere, vedere il retro copertina di questo manuale.
- Quando le due stringhe sono ordinate come in un elenco telefonico, l'ordine delle stringhe determina quale è la maggiore.



Copy File (COP) Synchronous Copy File (CPS)

Le istruzioni COP e CPS copiano il(i) valore(i) in Source nella Destination. Il valore Source rimane invariato.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source	**SINT** **INT** **DINT** **REAL** **stringa** struttura	tag	elemento iniziale da copiare **Importante:** per evitare risultati inaspettati gli operandi Source e Destination devono essere dello stesso tipo di dati
Destination	**SINT** **INT** **DINT** **REAL** **stringa** struttura	tag	elemento iniziale che Source deve sovrascrivere **Importante:** per evitare risultati inaspettati gli operandi Source e Destination devono essere dello stesso tipo di dati
Length	DINT	immediato tag	numero di elementi di destinazione da copiare

Testo strutturato

```
COP(Source,Dest,Length);
CPS(Source,Dest,Length);
```

Gli operandi sono gli stessi delle istruzioni COP e CPS in ladder.

Descrizione: Durante l'esecuzione delle istruzioni COP e CPS, altre azioni del Controllore possono provare ad interrompere l'operazione di copiatura e modificare i dati sorgente o destinazione:

Se Source o Destination è	E si desidera	Selezionare	Note
• tag prodotto • tag consumato • dati I/O • dati che un altro task può sovrascrivere	preservare i dati da modifiche durante l'operazione di copiatura	CPS	• I task che tentano di interrompere un'istruzione CPS vengono ritardati fino a quando l'istruzione è eseguita. • Per stimare il tempo di esecuzione di un'istruzione CPS, vedere *Sistema ControlLogix – Manuale dell'utente*, pubblicazione 1756-UM001.
	consentire la modifica dei dati durante l'operazione di copiatura	COP	
nessuno dei precedenti	→	COP	

Il numero di byte copiati è:

Conteggio byte = Length * (numero di byte nel tipo di dati della Destination)

ATTENZIONE

Se il numero di byte è superiore alla lunghezza di Source, per i rimanenti elementi verranno copiati dei dati imprevedibili.

IMPORTANTE

È ***necessario*** testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

Le istruzioni COP e CPS operano su dati di memoria contigui compiono una copia di memoria diretta da byte a byte. In alcuni casi, scrivono oltre la matrice in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

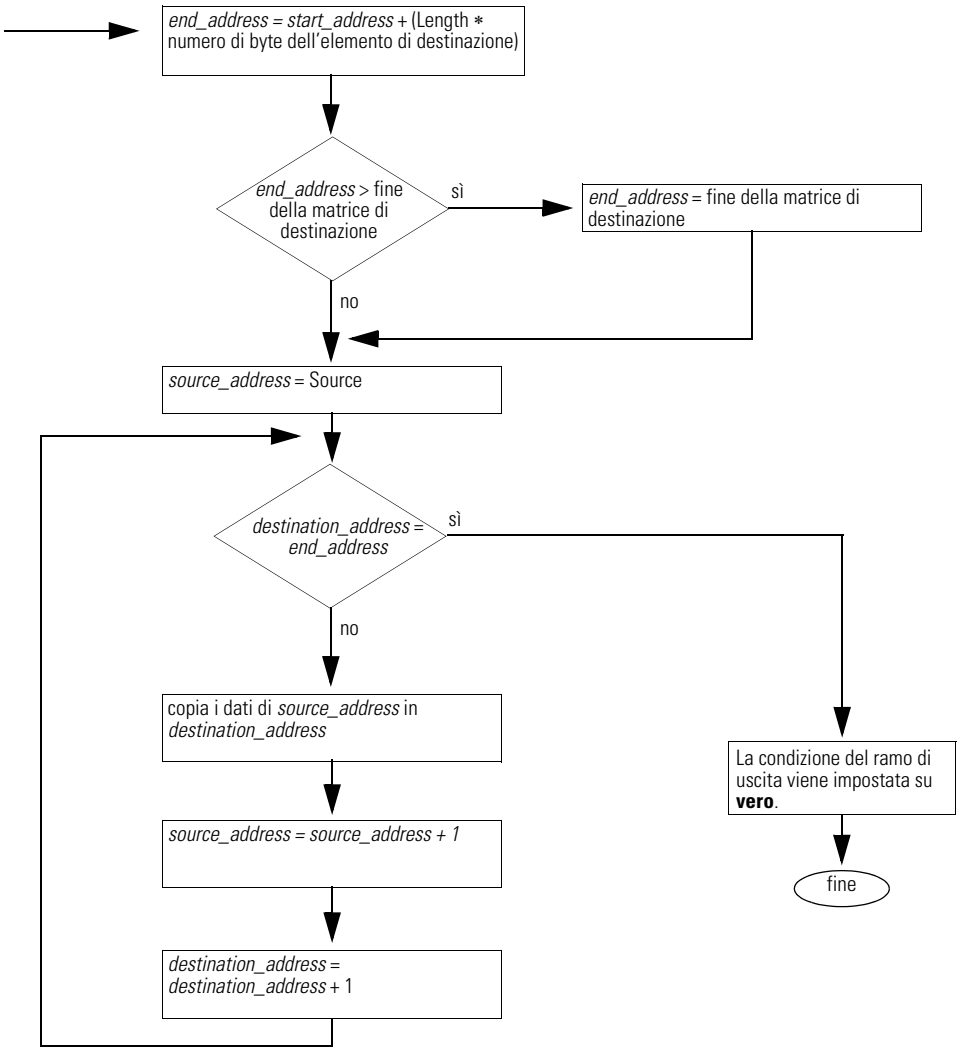
Se il tag è:	Allora
tipo di dati definito dall'utente	Se la lunghezza è eccessiva, l'istruzione scrive oltre la fine della matrice in altri membri del tag. Si ferma alla fine del tag. Non viene generato alcun errore grave.
tipo di dati NON definito dall'utente	Se la lunghezza è eccessiva, l'istruzione si arresta alla fine della matrice. Non viene generato alcun errore grave.

La lunghezza è eccessiva se supera il numero totale di elementi nella matrice Destination.

Indicatori di stato aritmetico: non influenzati

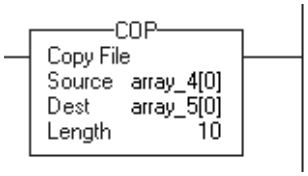
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni		
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio 1: Sia *array_4* che *array_5* sono dello stesso tipo di dati. Se è abilitata, l'istruzione COP copia i primi 10 elementi di *array_4* nei primi 10 elementi di *array_5*.

Ladder

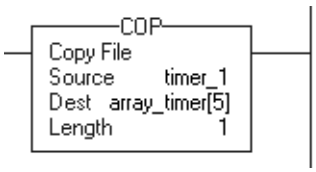


Testo strutturato

```
COP(array_4[0],array_5[0],10);
```

Esempio 2: Se è abilitata, l'istruzione COP copia la struttura *timer_1* nell'elemento 5 di *array_timer*. L'istruzione copia solamente una struttura in un elemento della matrice.

Ladder



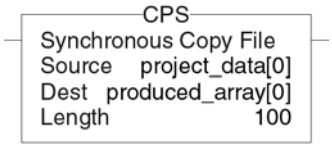
Testo strutturato

```
COP(timer_1,array_timer[5],1);
```

Esempio 3: *La matrice* project_data (100 elementi) memorizza una varietà di valori che si modificano in tempi differenti nell'applicazione. Per inviare un'immagine completa di *project_data* in un istante di tempo ad un altro Controllore, l'istruzione CPS copia *project_data* su *produced_array*.

- Mentre l'istruzione CPS copia i dati, nessun aggiornamento I/O o altro task può modificare i dati.
- Il tag *produced_array* produce i dati su una rete ControlNet ad uso di altri Controllore.
- Per usare la stessa immagine dei dati (cioè, una copia sincronizzata dei dati), il Controllore che usa i dati utilizza un'istruzione CPS per copiare i dati dal tag consumato a un altro tag per l'uso nell'applicazione.

Ladder



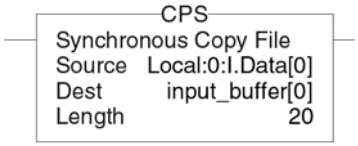
Testo strutturato

```
CPS(project_data[0],produced_array[0],100);
```

Esempio 4: *Local:0:I.Data* memorizza i dati di ingresso per la rete DeviceNet che è connessa al modulo 1756-DNB nello slot 0. Per sincronizzare gli ingressi con l'applicazione, l'istruzione CPS copia i dati di ingresso su *input_buffer*.

- Mentre l'istruzione CPS copia i dati, nessun aggiornamento I/O può modificare i dati.
- Quando viene lanciata, l'applicazione utilizza come suoi ingressi i dati di ingresso in *input_buffer*.

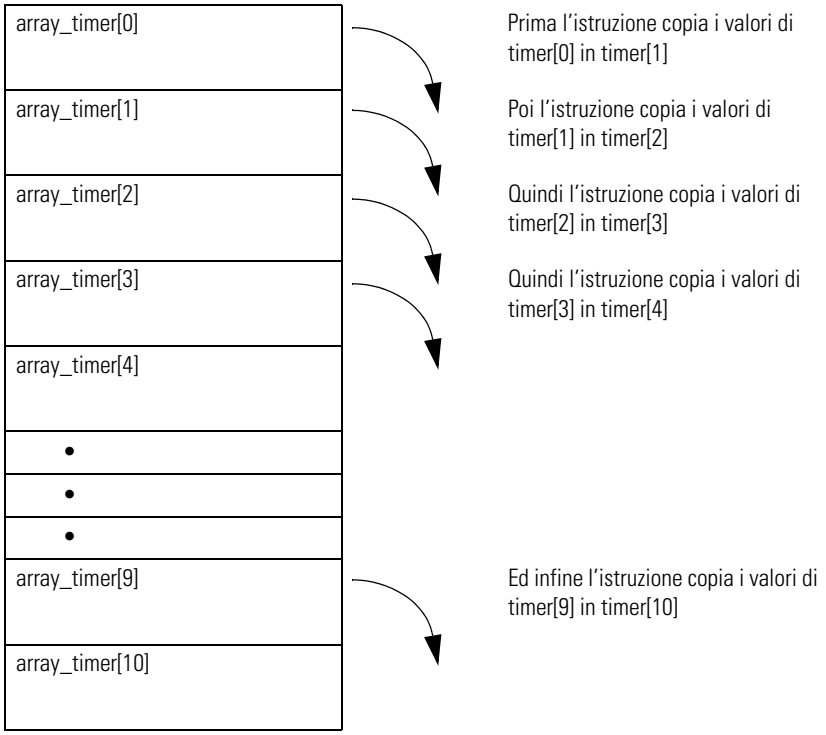
Ladder



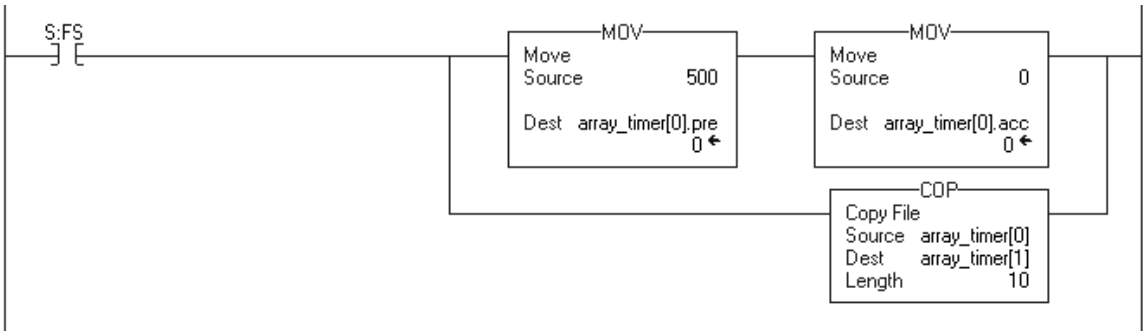
Testo strutturato

```
CPS(Local:0:I.Data[0],input_buffer[0],20);
```

Esempio 5: In questo esempio viene inizializzata una matrice di strutture timer. Se è abilitata, l'istruzione MOV inizializza i valori .PRE e .ACC del primo elemento di *array_timer*. Se è abilitata, l'istruzione COP copia un blocco contiguo di byte, a partire da *array_timer[0]*. La lunghezza è di nove strutture di timer.


array_timer[0]
array_timer[1]
array_timer[2]
array_timer[3]
array_timer[4]
array_timer[9]
array_timer[10]
Prima l'istruzione copia i valori di timer[0] in timer[1]
Poi l'istruzione copia i valori di timer[1] in timer[2]
Quindi l'istruzione copia i valori di timer[2] in timer[3]
Quindi l'istruzione copia i valori di timer[3] in timer[4]
Ed infine l'istruzione copia i valori di timer[9] in timer[10]

Ladder



Testo strutturato

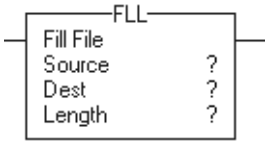
```
IF S:FS THEN

     array_timer[0].pre := 500;

     array_timer[0].acc := 0;

     COP(array_timer[0],array_timer[1],10);

END_IF;
```

File Fill (FLL)

L'istruzione FLL riempie gli elementi di una matrice con il valore Source. Il valore Source rimane invariato.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	**SINT** **INT** **DINT** **REAL**	immediato tag	elemento da copiare **Importante:** per evitare risultati inaspettati gli operandi Source e Destination devono essere dello stesso tipo di dati
Destination	**SINT** **INT** **DINT** **REAL** struttura	tag	elemento iniziale che Source deve sovrascrivere **Importante:** per evitare risultati inaspettati gli operandi Source e Destination devono essere dello stesso tipo di dati Il modo migliore per inizializzare una struttura è di utilizzare un'istruzione COP.
Length	DINT	immediato	numero di elementi da riempire

Testo strutturato

Il linguaggio in testo strutturato non prevede un'istruzione FLL, tuttavia è possibile ottenere lo stesso risultato utilizzando un'istruzione SIZE e un FOR...DO o altro costrutto loop.

```
SIZE(destination,0,length);
FOR position = 0 TO length-1 DO
    destination[position] := source;
END_FOR;
```

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi dei costrutti in testo strutturato.

Descrizione: Il numero di byte riempiti è:

Conteggio byte = Length * (numero di byte nel tipo di dati della Destination)

IMPORTANTE È ***necessario*** testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione FLL opera su una memoria di dati contigui. In alcuni casi, le istruzioni scrivono oltre la matrice in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

Se il tag è:	Allora
tipo di dati definito dall'utente	Se la lunghezza è eccessiva, l'istruzione scrive oltre la fine della matrice in altri membri del tag. Si ferma alla fine del tag. Non viene generato alcun errore grave.
tipo di dati NON definito dall'utente	Se la lunghezza è eccessiva, l'istruzione si arresta alla fine della matrice. Non viene generato alcun errore grave.

La lunghezza è eccessiva se supera il numero totale di elementi nella matrice Destination.

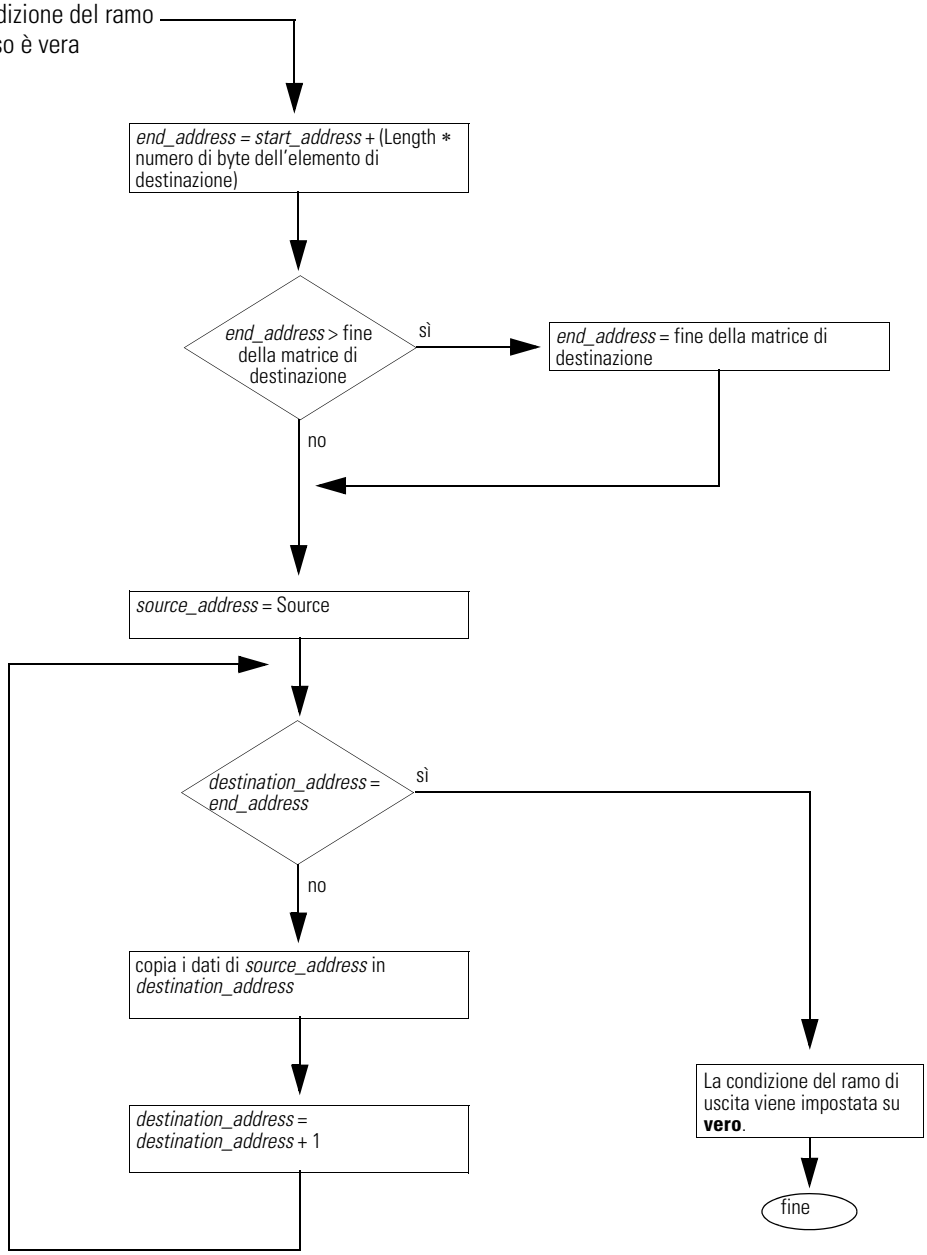
Per ottenere i migliori risultati i valori Source e Destination devono essere dello stesso tipo di dati. Se si desidera riempire una struttura, utilizzare l'istruzione COP (vedere esempio 3 a pagina 354). Se per Source e Destination si usano tipi di dati diversi, gli elementi di destinazione vengono riempiti con i valori Source convertiti.

Se il valore Source è	E la destinazione è	Il valore Source viene convertito in
SINT, INT, DINT o REAL	SINT	SINT
SINT, INT, DINT o REAL	INT	INT
SINT, INT, DINT o REAL	DINT	DINT
SINT, INT, DINT o REAL	REAL	REAL
SINT	struttura	SINT (non convertito)
INT	struttura	INT (non convertito)
DINT	struttura	DINT (non convertito)
REAL	struttura	REAL (non convertito)

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

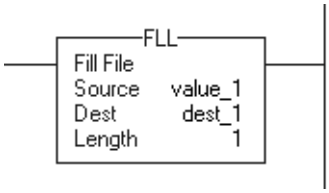
Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

end_address = *start_address* + (Length * numero di byte dell'elemento di destinazione)
end_address > fine della matrice di destinazione
sì
end_address = fine della matrice di destinazione
no
source_address = Source
destination_address = *end_address*
sì
no
copia i dati di *source_address* in *destination_address*
destination_address = *destination_address* + 1
La condizione del ramo di uscita viene impostata su **vero**.
fine

Esempio: Se è abilitata, l'istruzione FLL copia i dati di *value_1* in *dest_1*.

Ladder



Tipo di dati Source (*value_1*)	Valore Source (*value_1*)	Tipo di dati Destination (*dest_1*)	Valore Destination (*dest_1*) dopo FLL
SINT	16#80 (–128)	DINT	16#FFFF FF80 (–128)
DINT	16#1234 5678	SINT	16#78
SINT	16#01	REAL	1.0
REAL	2.0	INT	16#0002
SINT	16#01	TIMER	16#0101 0101 16#0101 0101 16#0101 0101
INT	16#0001	TIMER	16#0001 0001 16#0001 0001 16#0001 0001
DINT	16#0000 0001	TIMER	16#0000 0001 16#0000 0001 16#0000 0001

Testo strutturato

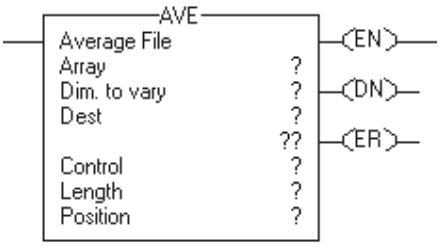
```
dest_1 := value_1;
```

File Average (AVE)

L'istruzione AVE calcola la media di un gruppo di valori.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Array	SINT	tag matrice	calcola la media dei valori di questa matrice
	INT		specifica il primo elemento del gruppo di elementi su cui calcolare la media
	DINT		**non** usare CONTROL.POS nell'indice
	REAL		
Dimension to vary	DINT	immediato	quale dimensione usare
		(0, 1, 2)	a seconda del numero delle dimensioni, l'ordine è
			array[dim_0,dim_1,dim_2]
			array[dim_0,dim_1]
			array[dim_0]
Destination	SINT	tag	risultato dell'operazione
	INT		
	DINT		
	REAL		
Control	CONTROL	tag	struttura di controllo per l'operazione
Length	DINT	immediato	numero di elementi della matrice su cui calcolare la media
Position	DINT	immediato	elemento corrente della matrice
			il valore iniziale generalmente è 0

Testo strutturato

Il linguaggio testo strutturato non prevede un'istruzione AVE, tuttavia è possibile ottenere lo stesso risultato utilizzando un'istruzione SIZE e un FOR...DO o altro costrutto loop.

```
SIZE(array,0,length);
sum := 0;
FOR position = 0 TO length-1 DO
    sum := sum + array[position];
END_FOR;
destination := sum / length;
```

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi dei costrutti in testo strutturato.

Struttura di CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione AVE è abilitata.
.DN	BOOL	Il bit di fine viene impostato quando l'istruzione ha operato sull'ultimo elemento di Array (.POS = .LEN).
.ER	BOOL	Il bit di errore viene impostato se l'istruzione genera un overflow. L'istruzione interrompe la sua esecuzione fino a quando il programma non azzera il bit .ER. La posizione dell'elemento che ha provocato l'overflow viene memorizzata nel valore .POS.
.LEN	DINT	La lunghezza specifica il numero di elementi della matrice su cui opera l'istruzione.
.POS	DINT	La posizione contiene la posizione dell'elemento corrente a cui l'istruzione accede.

Descrizione: L'istruzione AVE calcola la media di un gruppo di valori.

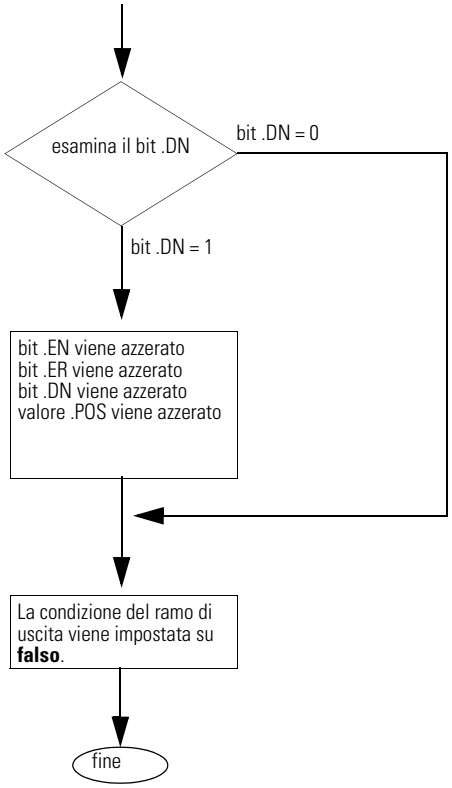
IMPORTANTE Assicurarsi che il valore Length non porti l'istruzione a superare la dimensione da variare specificata. In questo caso, la destinazione sarà errata.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
.POS < 0 oppure .LEN < 0	4	21
la dimensione da variare non esiste per la matrice specificata	4	20

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit .EN viene azzerato. Il bit .DN viene azzerato. Il bit .ER viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	
se la condizione del ramo di ingresso è vera	L'istruzione AVE calcola la media sommando tutti gli elementi specificati nella matrice e dividendo per il numero di elementi. Internamente, l'istruzione utilizza un'istruzione FAL per calcolare la media: Espressione = calcolo della media Modalità = ALL Per ulteriori informazioni sul funzionamento dell'istruzione FAL, vedere pagina 333.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

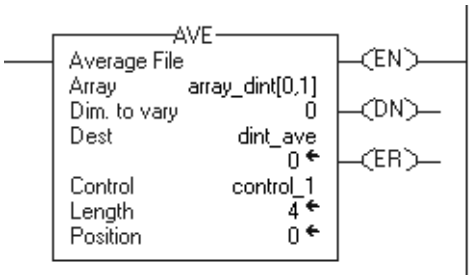
Esempio 1: Media di *array_dint*, ossia DINT[4,5].

indici	dimensione 1				
	0	1	2	3	4
dimensione 0 0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

$$AVE = \frac{19 + 14 + 9 + 4}{4} = \frac{46}{4} = 11.5$$

dint_ave = 12

Ladder



Testo strutturato

```
SIZE(array_dint,0,length);
sum := 0;
FOR position = 0 TO (length-1) DO
    sum := sum + array_dint[position];
END_FOR;
dint_ave := sum / length;
```

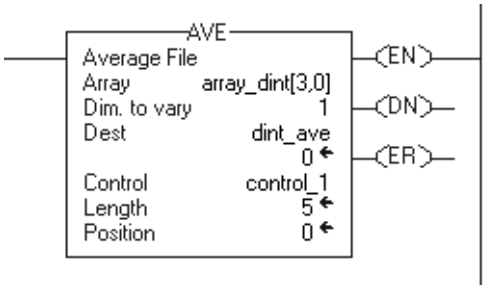
Esempio 2: Media di *array_dint*, ossia DINT[4,5].

indici	dimensione 1				
dimensione 0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

$$AVE = \frac{5 + 4 + 3 + 2 + 1}{5} = \frac{15}{5} = 3$$

$dint\_ave = 3$

Ladder



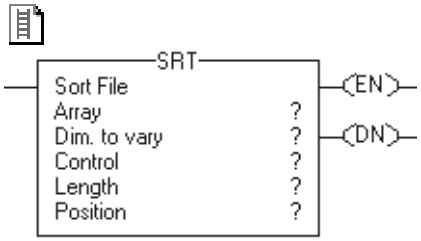
Testo strutturato

```
SIZE(array_dint,1,length);
sum := 0;
FOR position = 0 TO (length-1) DO
    sum := sum + array_dint[position];
END_FOR;
dint_ave := sum / length;
```

File Sort (SRT)

L'istruzione SRT ordina in modo crescente un gruppo di valori di una dimensione (Dim to vary) della Matrice.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Array	**SINT** **INT** **DINT** **REAL**	tag matrice	matrice da ordinare specifica il primo elemento del gruppo di elementi da ordinare **non** usare CONTROL.POS nell'indice
Dimension to vary	DINT	immediato (0, 1, 2)	quale dimensione usare a seconda del numero delle dimensioni, l'ordine è array[dim_0,dim_1,dim_2] array[dim_0,dim_1] array[dim_0]
Control	CONTROL	tag	struttura di controllo per l'operazione
Length	DINT	immediato	numero di elementi della matrice da ordinare
Position	DINT	immediato	elemento corrente della matrice il valore iniziale generalmente è 0

Testo strutturato

```
SRT(Array,Dimtovary,
    Control);
```

Gli operandi sono gli stessi dell'istruzione SRT in ladder. Tuttavia, i valori Length e Position vengono indicati accedendo ai membri .LEN e .POS della struttura di controllo invece di inserire il valore nella lista di operandi.

Struttura di CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione SRT è abilitata.
.DN	BOOL	Il bit di fine è impostato quando gli elementi specificati sono stati ordinati.
.ER	BOOL	Il bit di errore è impostato quando .LEN < 0 o quando .POS < 0. Una qualsiasi di queste condizioni genera anche un errore grave.
.LEN	DINT	La lunghezza specifica il numero di elementi della matrice su cui opera l'istruzione.
.POS	DINT	La posizione contiene la posizione dell'elemento corrente a cui l'istruzione accede.

Descrizione: L'istruzione SRT ordina in modo crescente un gruppo di valori di una dimensione (Dim da variare) della Matrice.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione SRT opera su una memoria contigua. In alcuni casi, l'istruzione cambia i dati in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

IMPORTANTE Assicurarsi che il valore Length non porti l'istruzione a superare la dimensione da variare specificata. Se ciò accadesse, si potrebbero verificare dei risultati inaspettati.

Questa è un'istruzione di transizione:

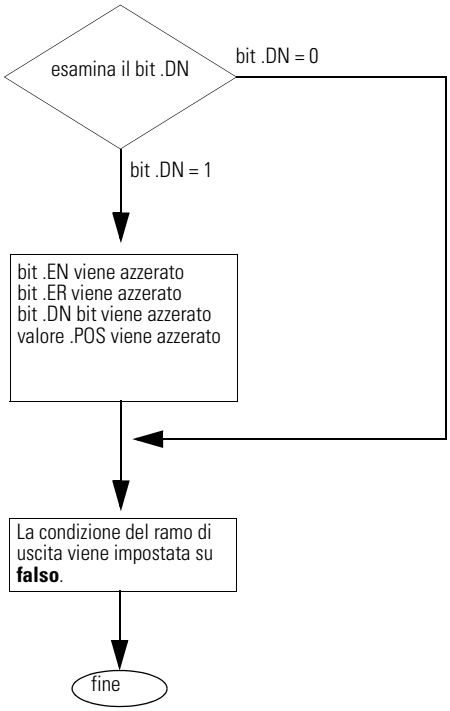
- In ladder, cambiare la condizione del ramo di ingresso da azzerata a impostata ogni volta che l'istruzione deve essere eseguita.
- In testo strutturato, condizionare l' istruzione in modo che venga eseguita solo su una transizione.
  Vedere Appendice C, Programmazione in testo strutturato.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
.POS < 0 oppure .LEN < 0	4	21
La dimensione da variare non esiste per la matrice specificata	4	20
L'istruzione tenta di avere accesso ai dati posti all'esterno dei limiti della matrice	4	20

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	Il bit .EN viene azzerato. Il bit .DN viene azzerato. Il bit .ER viene azzerato. La condizione del ramo di uscita viene impostata su falso.	Il bit .EN viene azzerato. Il bit .DN viene azzerato. Il bit .ER viene azzerato.
se la condizione del ramo di ingresso è falsa		na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione ordina gli elementi specificati della matrice in modo crescente.	L'istruzione ordina gli elementi specificati della matrice in modo crescente.
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio 1: Ordina *int* _array, ossia DINT[4,5].

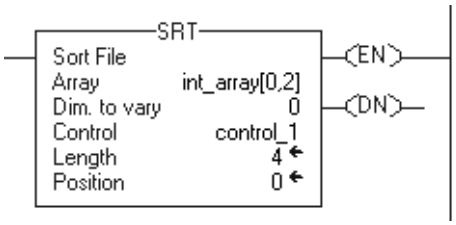
Prima

indici	dimensione 1				
dimensione 0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

Dopo

indici	dimensione 1				
dimensione 0	0	1	2	3	4
0	20	19	3	17	16
1	15	14	8	12	11
2	10	9	13	7	6
3	5	4	18	2	1

Ladder



Testo strutturato

```
control_1.LEN := 4;

control_1.POS := 0;

SRT(int_array[0,2],0,control_1);
```

Esempio 2: Ordina *int* _array, ossia DINT[4,5].

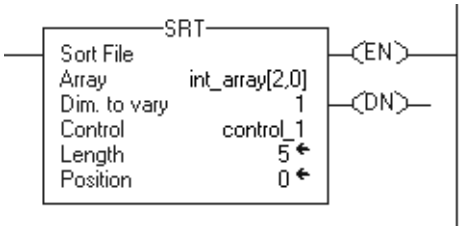
Prima

indici	dimensione 1				
dimensione 0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

Dopo

indici	dimensione 1				
dimensione 0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	6	7	8	9	10
3	5	4	3	2	1

Ladder



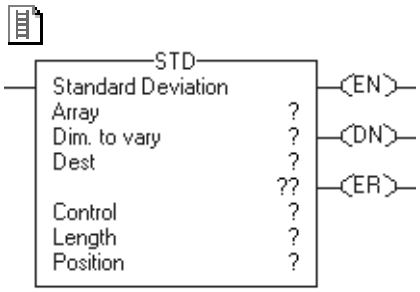
Testo strutturato

```
control_1.LEN := 5;

control_1.POS := 0;

SRT(int_array[2,0],1,control_1);
```

File Standard Deviation (STD)

L'istruzione STD calcola la deviazione standard di un insieme di valori di una dimensione della matrice e memorizza il risultato nella destinazione.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Array	SINT	tag matrice	calcola la deviazione standard dei valori di una matrice
	INT		specifica il primo elemento del gruppo di elementi da utilizzare per il calcolo della deviazione standard
	DINT		
	REAL		**non** usare CONTROL.POS nell'indice
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Dimension to vary	DINT	immediato	quale dimensione usare
		(0, 1, 2)	a seconda del numero delle dimensioni, l'ordine è
			array[dim_0,dim_1,dim_2]
			array[dim_0,dim_1]
			array[dim_0]
Destination	REAL	tag	risultato dell'operazione
Control	CONTROL	tag	struttura di controllo per l'operazione
Length	DINT	immediato	numero di elementi della matrice da utilizzare per il calcolo della deviazione standard
Position	DINT	immediato	elemento corrente della matrice
			il valore iniziale generalmente è 0

Struttura di CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione STD è abilitata.
.DN	BOOL	Il bit di fine viene impostato quando il calcolo è stato completato.
.ER	BOOL	Il bit di errore viene impostato se l'istruzione genera un overflow. L'istruzione interrompe la sua esecuzione fino a quando il programma non azzera il bit .ER. La posizione dell'elemento che ha provocato l'overflow viene memorizzata nel valore .POS.
.LEN	DINT	La lunghezza specifica il numero di elementi della matrice su cui opera l'istruzione.
.POS	DINT	La posizione contiene la posizione dell'elemento corrente a cui l'istruzione accede.

Testo strutturato

Il linguaggio testo strutturato non prevede un'istruzione STD, tuttavia è possibile ottenere lo stesso risultato utilizzando un'istruzione SIZE e un FOR...DO o altro costrutto loop.

```
SIZE(array,0,length);
sum := 0;
FOR position = 0 TO length-1 DO
    sum := sum + array[position];
END_FOR;
average := sum / length;
sum := 0;
FOR position = 0 TO length-1 DO
    sum := sum + ((array[position] – average)**2);
END_FOR;
destination := SQRT(sum /(length-1));
```

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi dei costrutti in testo strutturato.

Descrizione: La deviazione standard viene calcolata in base alla seguente formula:

$$\text{Deviazione standard} = \sqrt{\frac{\left(\sum_{i=1}^{N}\left[\langle X_{(start+i)} - AVE\rangle^2\right]\right)}{(N-1)}}$$

Dove:

- start = indice della dimensione da variare dell'operando della matrice
- x_i = elemento variabile della matrice
- N = numero di elementi specificati della matrice
- AVE = $\dfrac{\left(\sum_{i=1}^{N} x_{(start+i)}\right)}{N}$

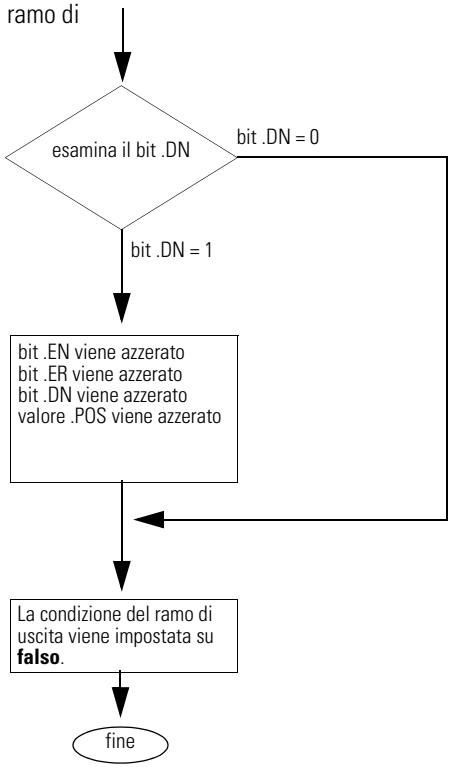
IMPORTANTE Assicurarsi che il valore Length non porti l'istruzione a superare la dimensione da variare specificata. In questo caso, la destinazione sarà errata.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
.POS < 0 oppure .LEN < 0	4	21
la dimensione da variare non esiste per la matrice specificata	4	20

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit .EN viene azzerato. Il bit .DN viene azzerato. Il bit .ER viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	
se la condizione del ramo di ingresso è vera	L'istruzione STD calcola la deviazione standard degli elementi specificati. Internamente, l'istruzione utilizza un'istruzione FAL per calcolare la media: Espressione = calcolo della deviazione standard Modalità = ALL Per ulteriori informazioni sul funzionamento dell'istruzione FAL, vedere pagina 333.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio 1: Calcolo della deviazione standard di *dint_array*, ossia DINT[4,5].

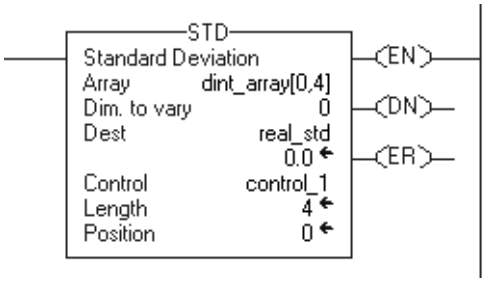
indici	dimensione 1				
dimensione 0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

$$AVE = \frac{16 + 11 + 6 + 1}{4} = \frac{34}{4} = 8.5$$

$$STD = \sqrt{\frac{\langle 16 - 8.5\rangle^2 + \langle 11 - 8.5\rangle^2 + \langle 6 - 8.5\rangle^2 + \langle 1 - 8.5\rangle^2}{\langle 4 - 1\rangle}} = 6.454972$$

real_std = 6,454972

Ladder



Testo strutturato

```
SIZE(dint_array,0,length);
sum := 0;
FOR position = 0 TO (length-1) DO
    sum := sum + dint_array[position];
END_FOR;
average := sum / length;
sum := 0;
FOR position = 0 TO (length-1) DO
    sum := sum + ((dint_array[position] – average)**2);
END_FOR;
real_std := SQRT(sum /(length-1));
```

Esempio 2: Calcolo della deviazione standard di *dint_array*, ossia DINT[4,5].

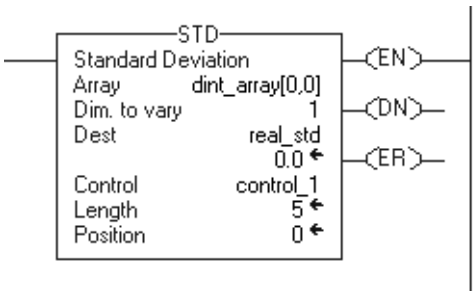
indici	dimensione 1				
dimensione 0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

$$AVE = \frac{20 + 19 + 18 + 17 + 16}{5} = \frac{90}{5} = 18$$

$$STD = \sqrt{\frac{\langle 20 - 18\rangle^2 + \langle 19 - 18\rangle^2 + \langle 18 - 18\rangle^2 + \langle 17 - 18\rangle^2 + \langle 16 - 18\rangle^2}{\langle 5 - 1\rangle}} = 1.581139$$

real_std = 1,581139

Ladder



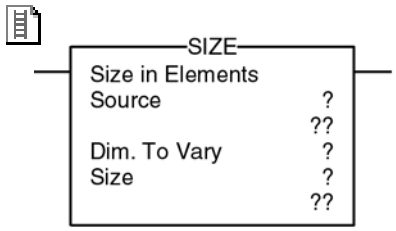
Testo strutturato

```
SIZE(dint_array,1,length);
sum := 0;
FOR position = 0 TO (length-1) DO
    sum := sum + dint_array[position];
END_FOR;
average := sum / length;
sum := 0;
FOR position = 0 TO (length-1) DO
    sum := sum + ((dint_array[position] – average)**2);
END_FOR;
real_std := SQRT(sum /(length-1));
```

Size In Elements (SIZE)

L'istruzione SIZE trova la misura di una dimensione di una matrice.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source	SINT INT DINT REAL struttura stringa	tag matrice	matrice sulla quale l'istruzione deve operare
Dimension to Vary	DINT	immediato (0, 1, 2)	dimensione da Usare **Per la dimensione di** / **Digitare** prima dimensione / 0 seconda dimensione / 1 terza dimensione / 2
Size	SINT INT DINT REAL	tag	tag per memorizzare il numero di elementi nella dimensione specificata della matrice

Testo strutturato

```
SIZE(Source,Dimtovary,Size);
```

Gli operandi sono gli stessi dell'istruzione SIZE in ladder.

Descrizione: L'istruzione SIZE trova il numero di elementi (size) nella dimensione designata della matrice Source e posiziona il risultato nell'operando Size.

- L'istruzione trova la misura di una dimensione di una matrice.
- L'istruzione opera su una:
  - matrice
  - matrice in una struttura
  - matrice che è parte di una matrice più grande

Indicatori di stato aritmetico: non influenzati

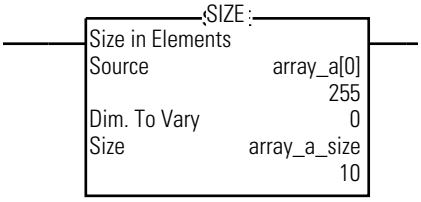
Condizioni di errore: nessuna.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione trova la misura (size) di una dimensione di una matrice.	L'istruzione trova la misura (size) di una dimensione di una matrice.
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio 1: Trova il numero di elementi in dimensione 0 (prima dimensione) di *array_a*. Memorizza la dimensione in *array_a_size*. In questo esempio, dimensione 0 di *array_a* ha 10 elementi.

Ladder

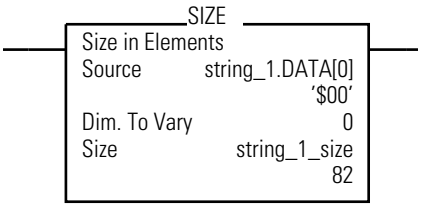


Testo strutturato

```
SIZE(array_a,0,array_a_size);
```

Esempio 2: Trova il numero di elementi nel membro DATA di *string_1*, che è una stringa. Memorizza la dimensione in *string_1_size*. In questo esempio, il membro DATA di *string_1* ha 82 elementi. (La stringa usa il tipo di dati STRINGA di default.) Siccome ciascun elemento contiene un carattere, *string_1* può contenere fino a 82 caratteri.

Ladder

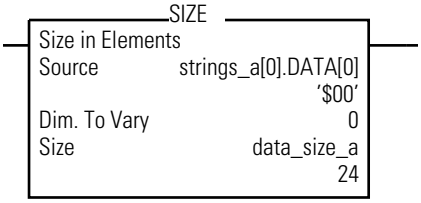


Testo strutturato

```
SIZE(string_1.DATA[0],0,string_1_size);
```

Esempio 3: *Strings_a* è una matrice di strutture stringa. L'istruzione SIZE trova il numero di elementi nel membro DATA della struttura stringa e memorizza la dimensione in *data_size_a*. In questo esempio, il membro DATA ha 24 elementi. (La struttura stringa ha una lunghezza definita dall'utente di 24.)

Ladder



Testo strutturato

```
SIZE(strings_a[0].DATA[0],0,data_size_a);
```

Note:

Istruzioni di matrici (File)/Scorrimento
(BSL, BSR, FFL, FFU, LFL, LFU)

Introduzione

Utilizzare le istruzioni di matrice (file)/scorrimento per modificare la posizione dei dati all'interno delle matrici.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
Caricare, spostare e scaricare bit da una matrice di bit, un bit alla volta.	BSL	ladder	380
	BSR	ladder	384
Caricare e scaricare valori nello stesso ordine.	FFL	ladder	388
	FFU	ladder	394
Caricare e scaricare valori nell'ordine inverso.	LFL	ladder	400
	LFU	ladder	406

È possibile mischiare vari tipi di dati, ma si potrebbero verificare errori di precisione e di arrotondamento.

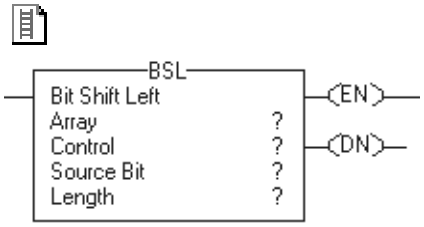
Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimali, in genere DINT o REAL.

Bit Shift Left (BSL)

L'istruzione BSL sposta i bit specificati della matrice di una posizione verso sinistra.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Array	DINT	tag matrice	matrice da modificare specifica il primo elemento del gruppo di elementi **non** usare CONTROL.POS nell'indice
Control	CONTROL	tag	struttura di controllo per l'operazione
Source bit	BOOL	tag	bit da spostare
Length	DINT	immediato	numero di bit della matrice da spostare

Struttura CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione BSL è abilitata.
.DN	BOOL	Il bit di fine è impostato per indicare che i bit sono stati spostati di una posizione a sinistra.
.UL	BOOL	Il bit di scaricamento è l'uscita dell'istruzione. Il bit .UL memorizza lo stato del bit che è stato spostato fuori dell'intervallo dei bit.
.ER	BOOL	Il bit di errore viene impostato quando .LEN < 0.
.LEN	DINT	La lunghezza specifica il numero di bit della matrice da spostare.

Descrizione: Se è abilitata, l'istruzione scarica il più alto dei bit specificati al bit .UL, fa scorrere i restanti bit di una posizione a sinistra e carica il Source bit nel bit 0 della matrice.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

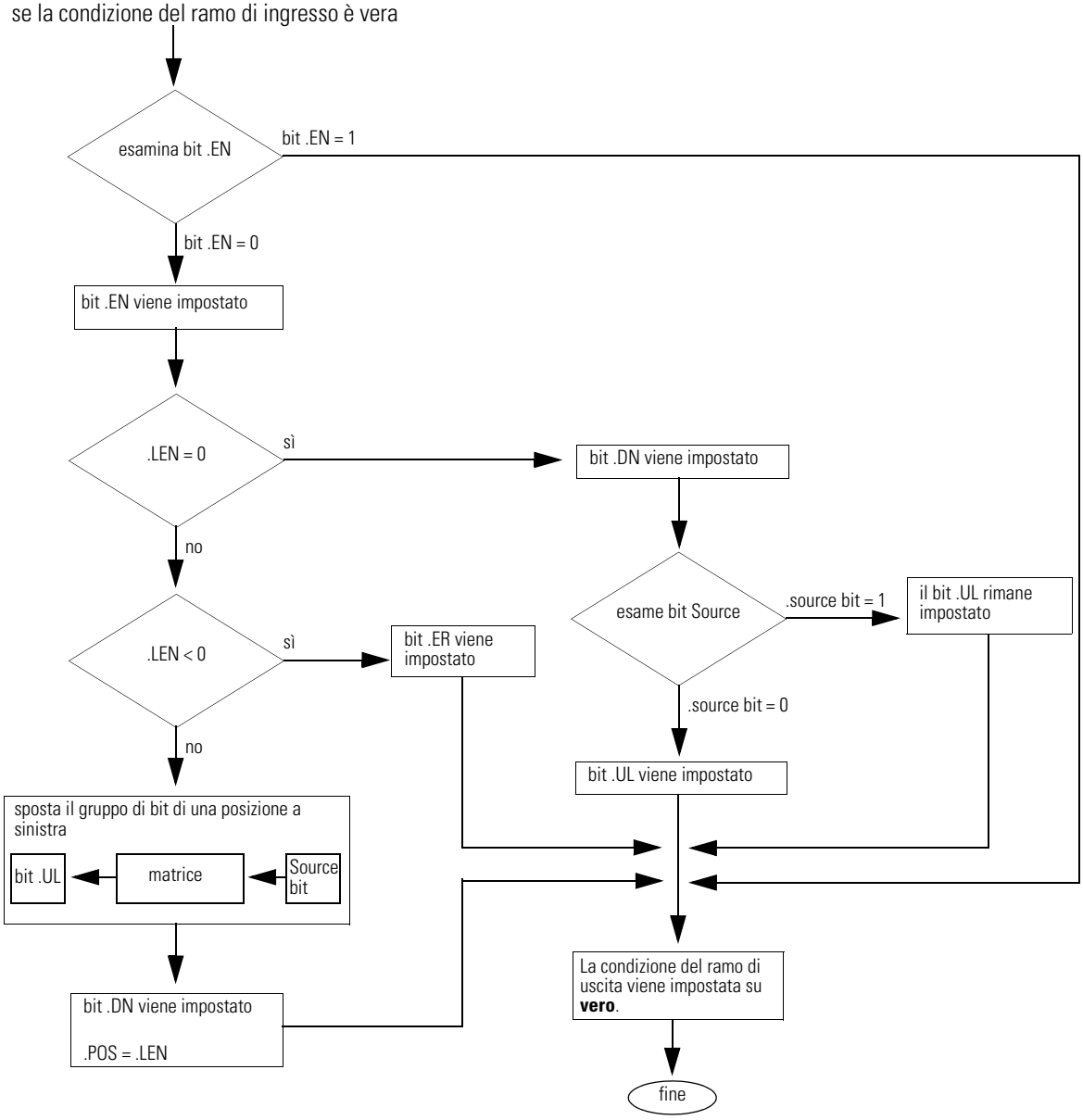
L'istruzione BSL opera su una memoria contigua. In alcuni casi, le istruzioni spostano i bit oltre la matrice in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

Indicatori di stato aritmetico: non influenzati

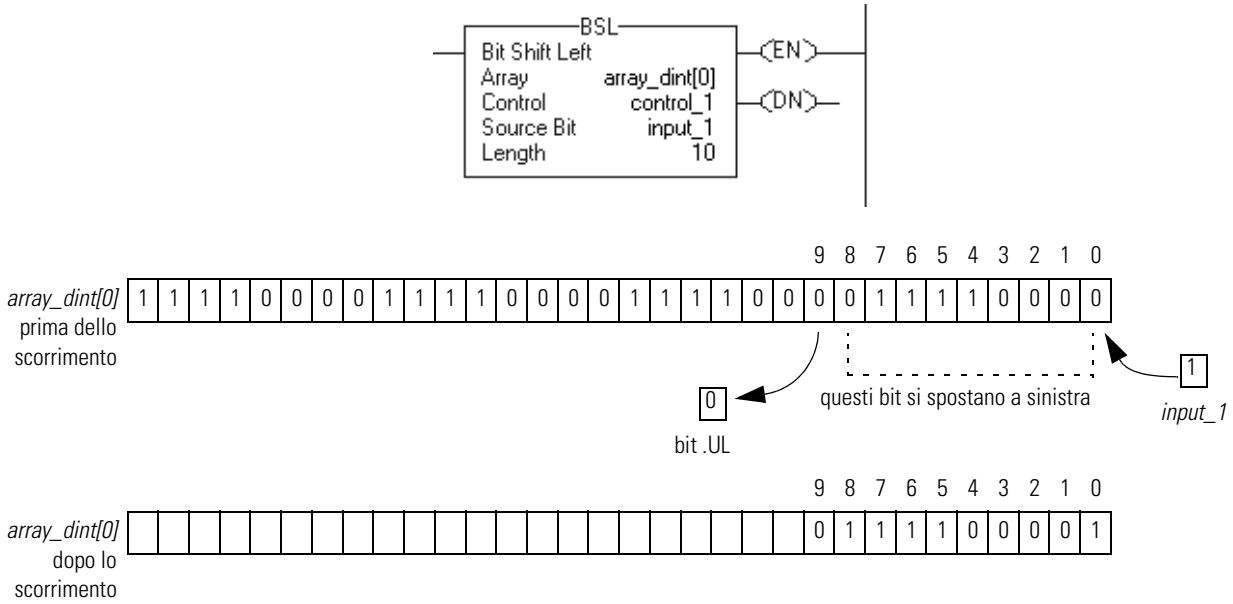
Condizioni di errore: nessuna

Esecuzione:

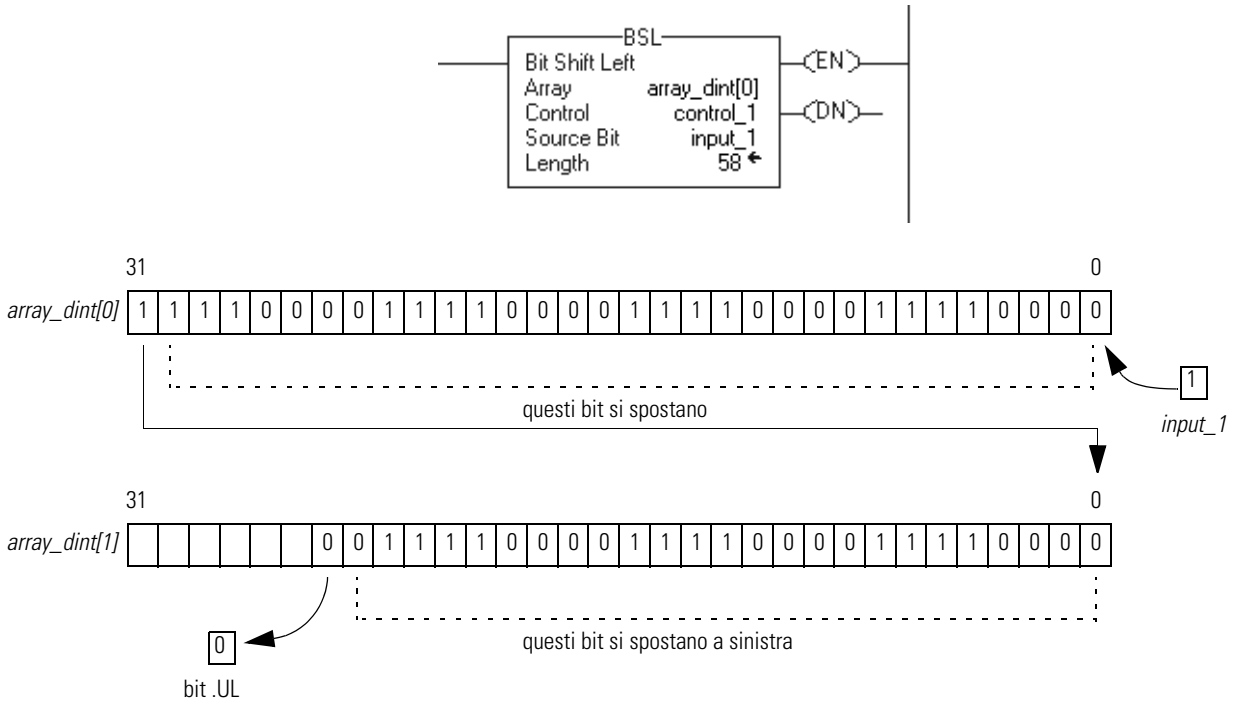
Condizione	Azione ladder
durante la prescansione	Il bit .EN viene azzerato.
	Il bit .DN viene azzerato.
	Il bit .ER viene azzerato.
	Il valore .POS viene azzerato.
	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit .EN viene azzerato.
	Il bit .DN viene azzerato.
	Il bit .ER viene azzerato.
	Il valore .POS viene azzerato.
	La condizione del ramo di uscita viene impostata su falso.

Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio 1: Se è abilitata, l'istruzione BSL inizia dal bit 0 di *array_dint[0]*. L'istruzione scarica *array_dint[0].9* nel bit .UL, sposta i rimanenti bit e carica *input_1* in *array_dint[0].0*. I valori dei bit rimanenti (10–31) non sono validi.


BSL
Bit Shift Left
Array array_dint[0]
Control control_1
Source Bit input_1
Length 10
EN
DN
9 8 7 6 5 4 3 2 1 0
array_dint[0]
prima dello
scorrimento
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
0
bit .UL
questi bit si spostano a sinistra
1
input_1
9 8 7 6 5 4 3 2 1 0
array_dint[0]
dopo lo
scorrimento
0 1 1 1 1 0 0 0 0 1

Esempio 2: Se è abilitata, l'istruzione BSL inizia dal bit 0 di *array_dint[0]*. L'istruzione scarica *array_dint[0].1* nel bit .UL, sposta i rimanenti bit e carica *input_1* in *array_dint[0].0*. I valori dei bit rimanenti (31–26 in *array_dint[1]*) non sono validi. Si noti come *array_dint[0].31* si sposti tra le parole fino a *array_dint[1].0*.

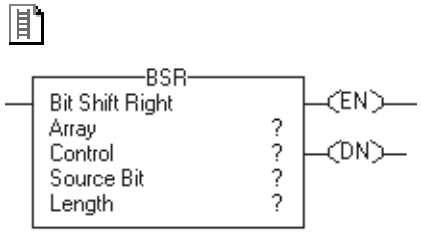

BSL
Bit Shift Left
Array array_dint[0]
Control control_1
Source Bit input_1
Length 58
EN
DN
31 0
array_dint[0]
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
questi bit si spostano
1
input_1
31 0
array_dint[1]
0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
0
bit .UL
questi bit si spostano a sinistra

Bit Shift Right (BSR)

L'istruzione BSR sposta i bit specificati della matrice di una posizione verso destra.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Array	DINT	tag matrice	matrice da modificare specifica l'elemento da cui iniziare lo scorrimento **non** usare CONTROL.POS nell'indice
Control	CONTROL	tag	struttura di controllo per l'operazione
Source bit	BOOL	tag	bit da spostare
Length	DINT	immediato	numero di bit della matrice da spostare

Struttura CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione BSR è abilitata.
.DN	BOOL	Il bit di fine è impostato per indicare che i bit sono stati spostati di una posizione a destra.
.UL	BOOL	Il bit di scaricamento è l'uscita dell'istruzione. Il bit .UL memorizza lo stato del bit che è stato spostato fuori dell'intervallo dei bit.
.ER	BOOL	Il bit di errore viene impostato quando .LEN < 0.
.LEN	DINT	La lunghezza specifica il numero di bit della matrice da spostare.

Descrizione: Se è abilitata, l'istruzione scarica il valore del bit 0 della matrice nel bit .UL, sposta i restanti bit di una posizione a destra e carica il bit Source nel più alto dei bit specificati.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

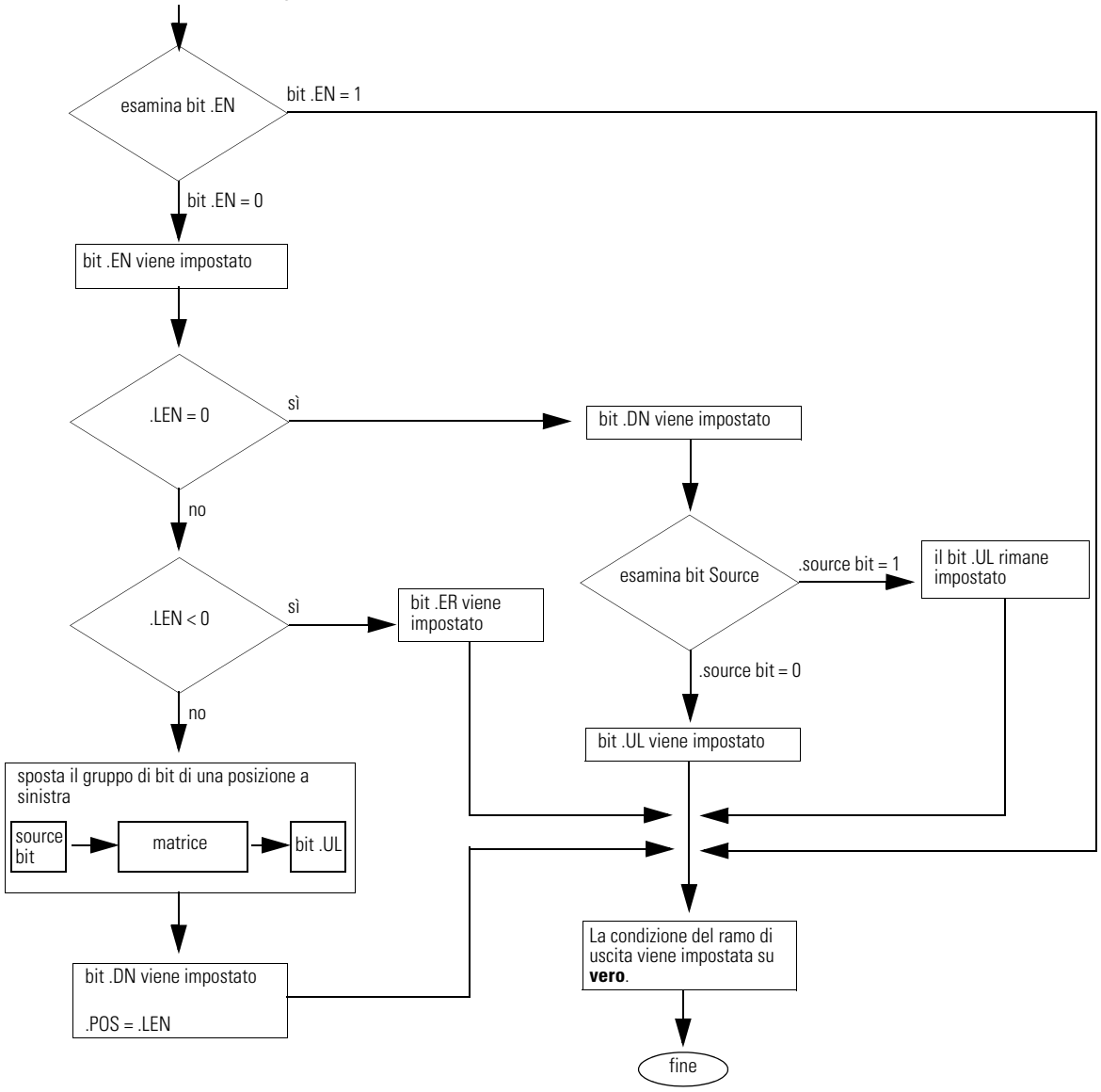
L'istruzione BSR opera su una memoria contigua. In alcuni casi, l'istruzione cambia i bit in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

Indicatori di stato aritmetico: non influenzati

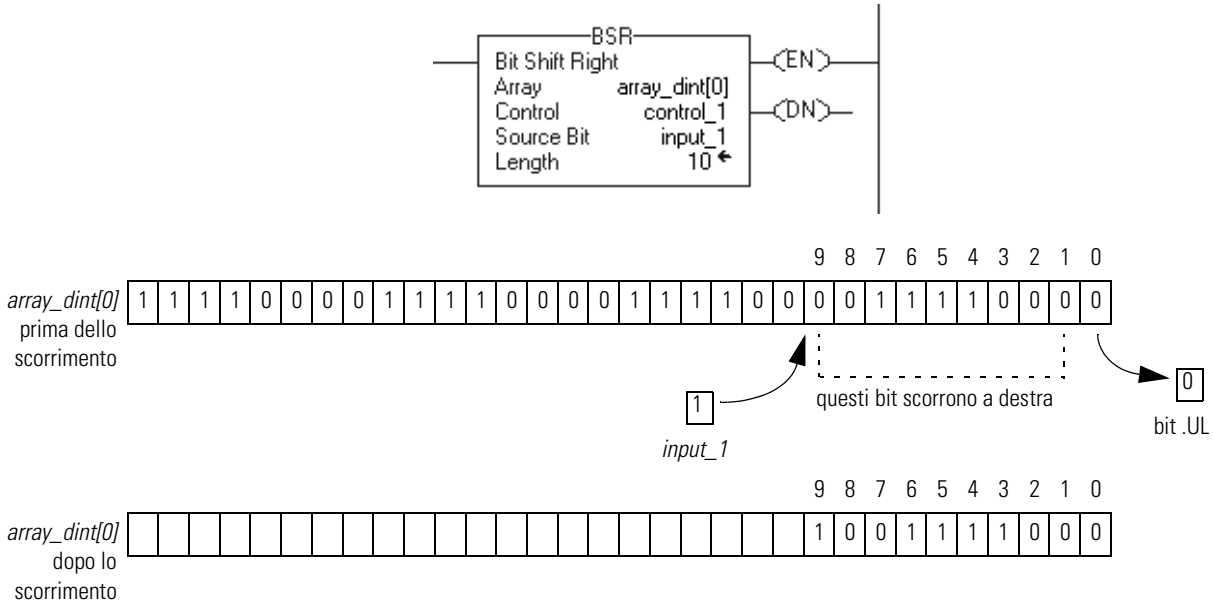
Condizioni di errore: nessuna

Esecuzione:

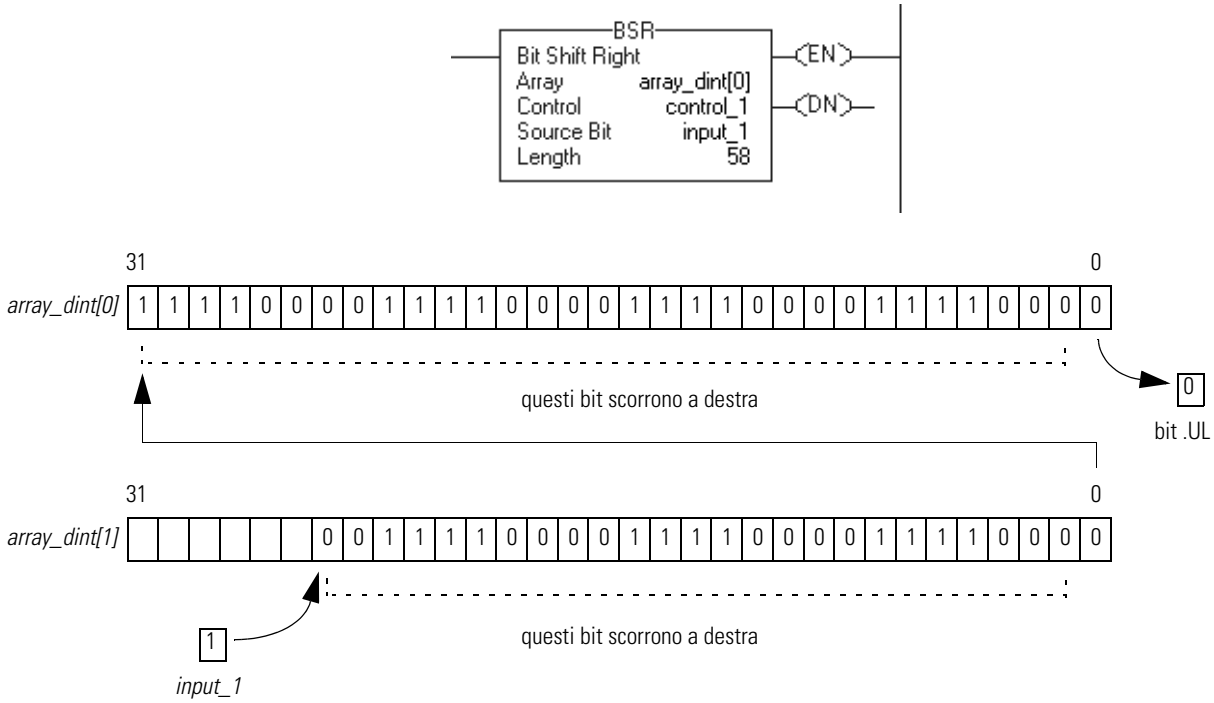
Condizione	Azione ladder
durante la prescansione	Il bit .EN viene azzerato. Il bit .DN viene azzerato. Il bit .ER viene azzerato. Il valore .POS viene azzerato. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit .EN viene azzerato. Il bit .DN viene azzerato. Il bit .ER viene azzerato. Il valore .POS viene azzerato. La condizione del ramo di uscita viene impostata su falso.

Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio 1: Se è abilitata, l'istruzione BSR inizia dal bit 9 di *array_dint[0]*. L'istruzione scarica *array_dint[0].0* nel bit .UL, sposta i rimanenti bit a destra e carica *input_1* in *array_dint[0].9*. I valori dei bit rimanenti (10–31) non sono validi.


BSR
Bit Shift Right
Array array_dint[0]
Control control_1
Source Bit input_1
Length 10
EN
DN
9 8 7 6 5 4 3 2 1 0
array_dint[0]
prima dello
scorrimento
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
questi bit scorrono a destra
0
bit .UL
1
input_1
9 8 7 6 5 4 3 2 1 0
array_dint[0]
dopo lo
scorrimento
1 0 0 1 1 1 1 0 0 0

Esempio 2: Se è abilitata, l'istruzione BSR inizia dal bit 25 di *array_dint[1]*. L'istruzione scarica *array_dint[0].0* nel bit .UL, sposta i rimanenti bit a destra e carica *input_1* in *array_dint[1].25*. I valori dei bit rimanenti (31–26 in *array_dint[1]*) non sono validi. Si noti come *array_dint[1].0* scorra attraverso le parole in *array_dint[0].31*.

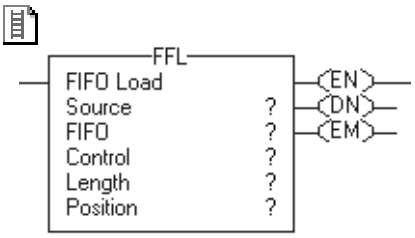

BSR
Bit Shift Right
Array array_dint[0]
Control control_1
Source Bit input_1
Length 58
EN
DN
31
0
array_dint[0]
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
questi bit scorrono a destra
0
bit .UL
31
0
array_dint[1]
0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
questi bit scorrono a destra
1
input_1

FIFO Load (FFL)

L'istruzione FFL copia il valore di Origine nel FIFO.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	dati da memorizzare nel FIFO
	INT	tag	
	DINT		
	REAL		
	stringa		
	struttura		
	Source viene convertito nel tipo di dati del tag matrice. Un intero più piccolo viene convertito in un intero più grande tramite estensione del segno.		
FIFO	SINT	tag matrice	FIFO da modificare
	INT		specifica il primo elemento del FIFO
	DINT		**non** usare CONTROL.POS nell'indice
	REAL		
	stringa		
	struttura		
Control	CONTROL	tag	struttura di controllo per l'operazione
			in genere, utilizzare lo stesso CONTROL del FFU associato
Length	DINT	immediato	numero massimo di elementi che il FIFO può contenere contemporaneamente
Position	DINT	immediato	posizione successiva nel FIFO in cui l'istruzione carica i dati
			il valore iniziale generalmente è 0

Se come tipo di dati per l'operando Source o FIFO si utilizza una struttura definita dall'utente, utilizzare la stessa struttura per entrambi gli operandi.

Struttura di controllo

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione FFL è abilitata.
.DN	BOOL	Il bit di fine è impostato per indicare che il FIFO è pieno (.POS = .LEN). Il bit .DN impedisce il caricamento del FIFO finché .POS < .LEN.
.EM	BOOL	Il bit di vuoto indica che il FIFO è vuoto. Se .LEN ≤ 0 o .POS < 0, vengono impostati sia il .EM sia il bit .DN.
.LEN	DINT	La lunghezza specifica il numero massimo di elementi che il FIFO può contenere contemporaneamente.
.POS	DINT	La posizione identifica la posizione del FIFO in cui l'istruzione caricherà il valore successivo.

Descrizione: Utilizzare l'istruzione FFL con l'istruzione FFU per memorizzare ed estrarre dati nell'ordine primo ad entrare/primo ad uscire. Se sono usate in coppia, le istruzioni FFL e FFU formano un registro a scorrimento asincrono.

In genere Source e FIFO sono dello stesso tipo di dati.

Se è abilitata, l'istruzione FFL carica il valore della Source nella posizione del FIFO identificata dal valore .POS. L'istruzione carica un valore ogni volta che l'istruzione è abilitata fino a quando il FIFO non è pieno.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione FFL opera su una memoria contigua. In alcuni casi, le istruzioni caricano i dati oltre la matrice in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

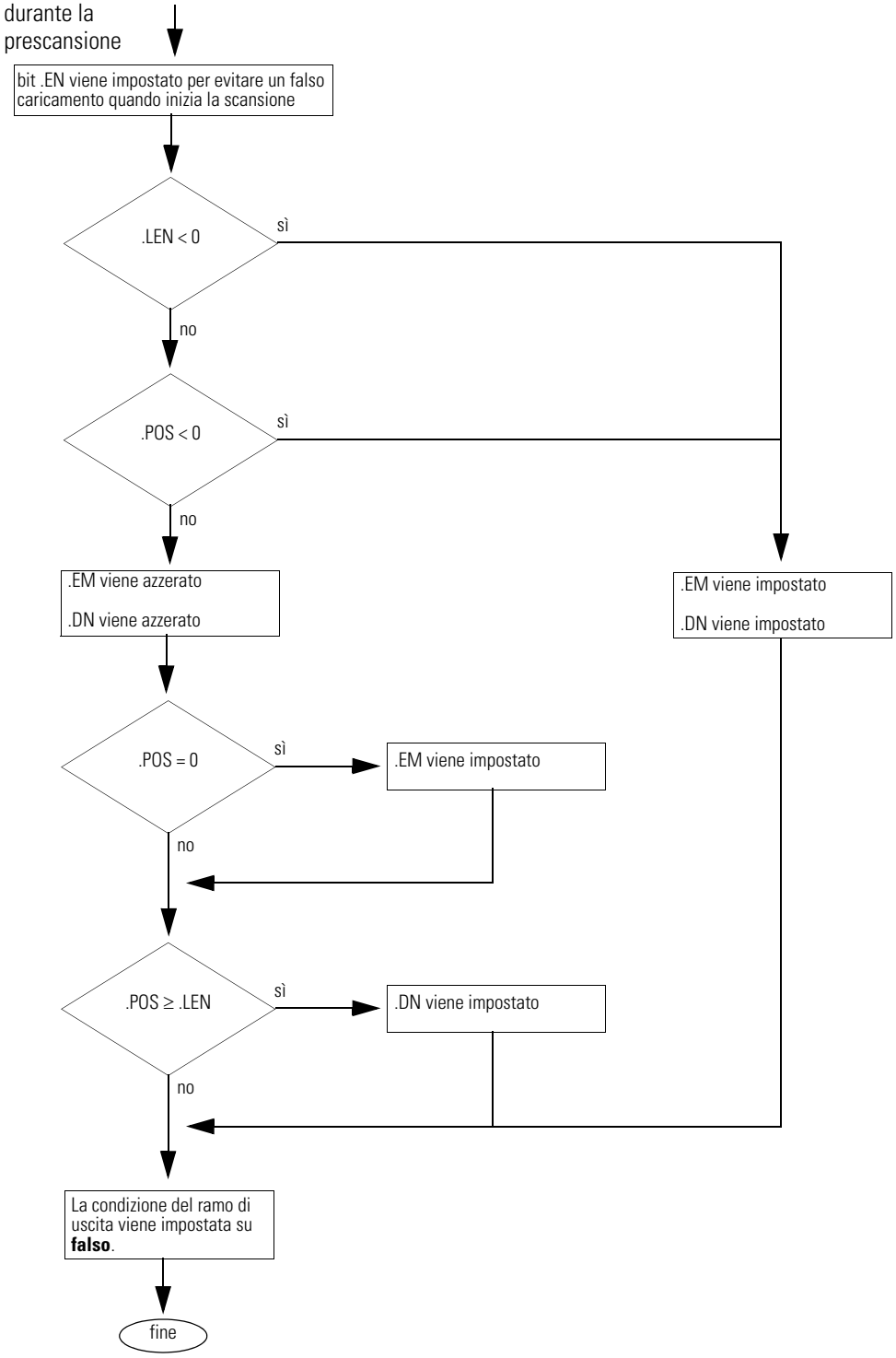
Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

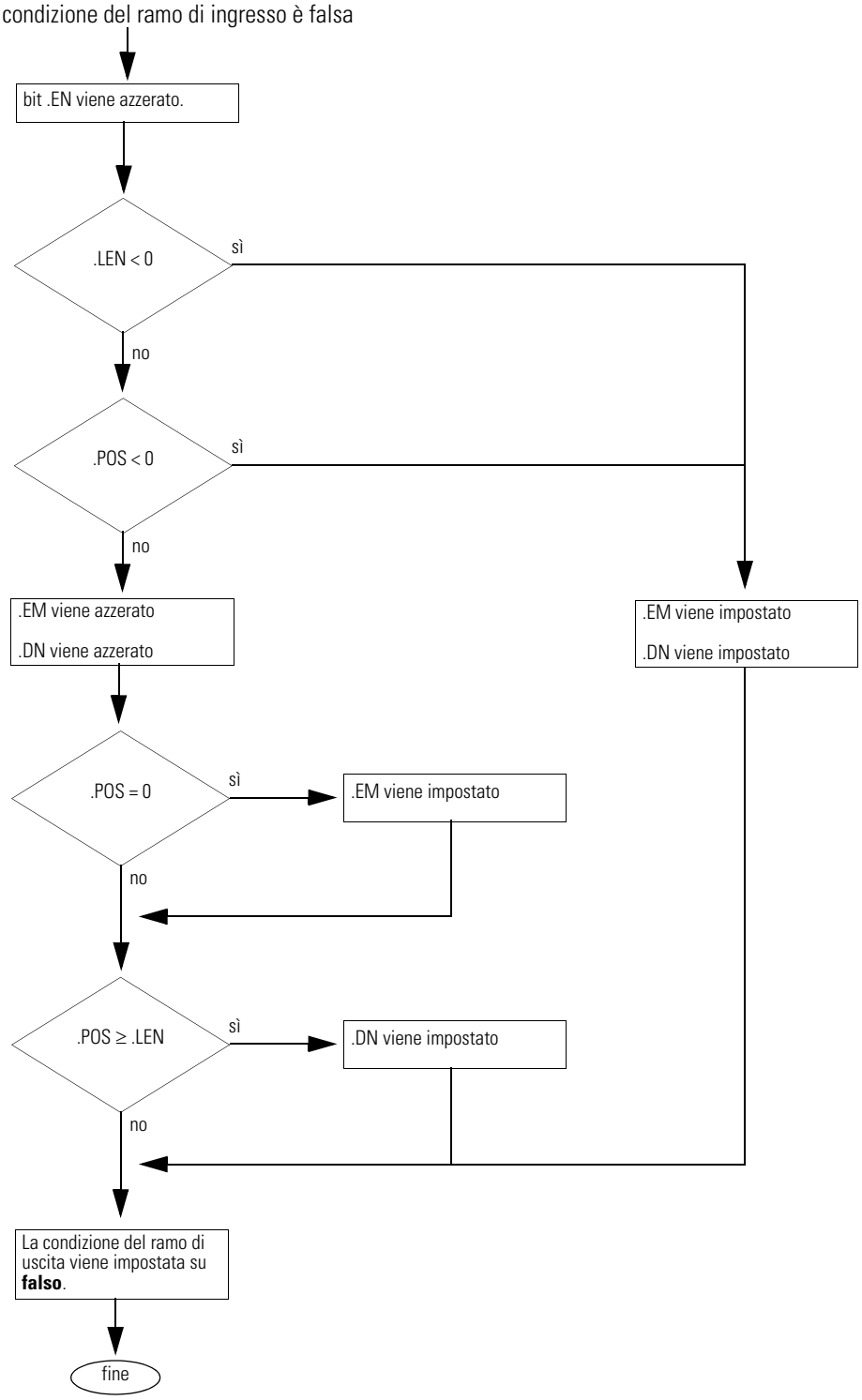
Si verifica un errore grave se	Tipo errore	Codice errore
(elemento iniziale + .POS) > dimensione matrice FIFO	4	20

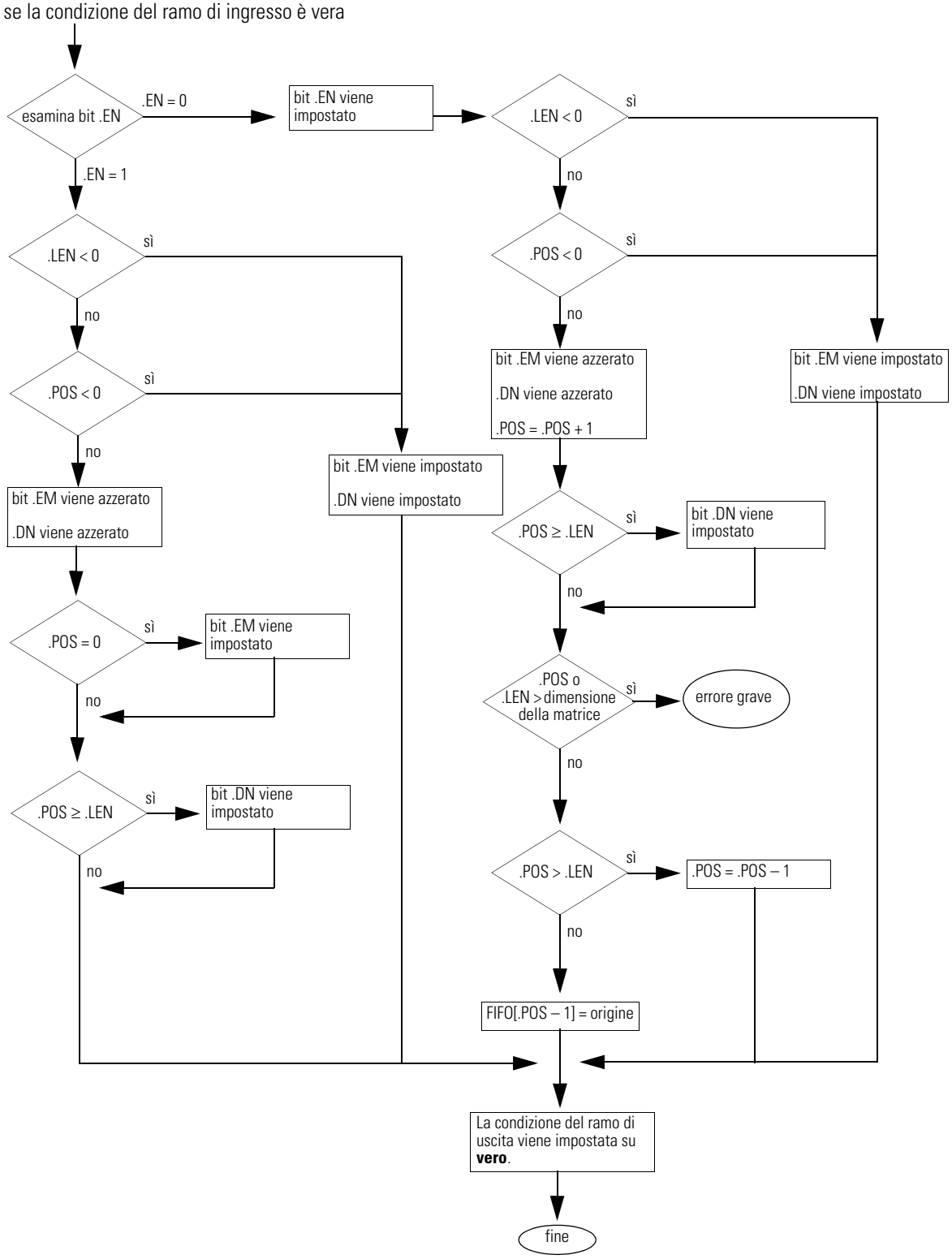
Esecuzione:

Condizione	Azione ladder

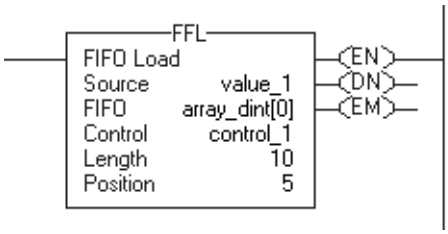


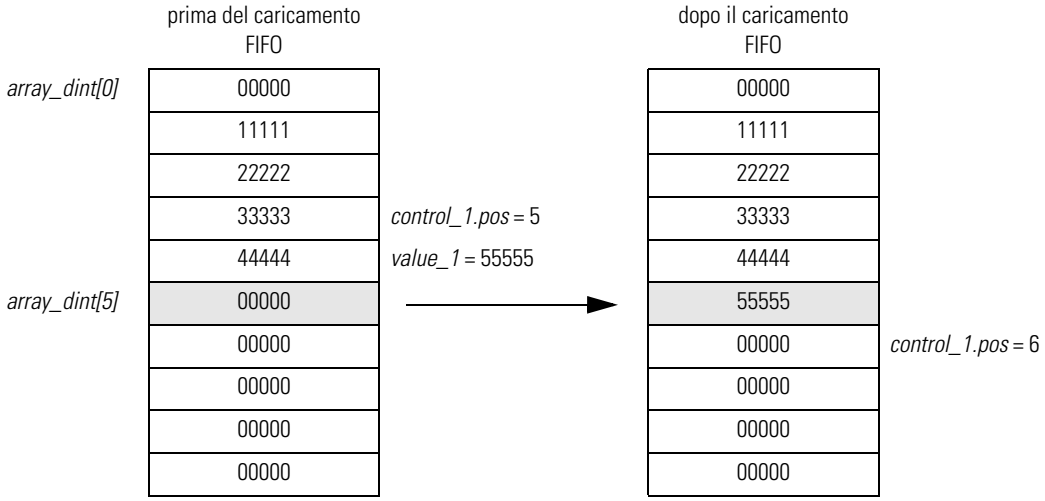
Condizione	Azione ladder
se la condizione del ramo di ingresso è falsa	



Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione FFL carica *value_1* nella posizione successiva della FIFO, che in questo esempio è *array_dint[5]*.



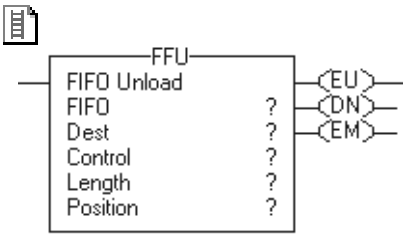


FIFO Unload (FFU)

L'istruzione FFU scarica il valore dalla posizione 0 (prima posizione) del FIFO e lo memorizza nella destinazione. I restanti dati del FIFO scorrono di una posizione.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
FIFO	SINT	tag matrice	FIFO da modificare
	INT		specifica il primo elemento del FIFO
	DINT		**non** usare CONTROL.POS nell'indice
	REAL		
	stringa		
	struttura		
Destination	SINT	tag	valore che esce dal FIFO
	INT		
	DINT		
	REAL		
	stringa		
	struttura		
	Il valore Destination viene convertito nel tipo di dati del tag Destination. Un intero più piccolo viene convertito in un intero più grande tramite estensione del segno.		
Control	CONTROL	tag	struttura di controllo per l'operazione in genere, utilizzare lo stesso CONTROL del FFL associato
Length	DINT	immediato	numero massimo di elementi che il FIFO può contenere contemporaneamente
Position	DINT	immediato	posizione successiva nel FIFO da cui l'istruzione scarica i dati il valore iniziale generalmente è 0

Se come tipo di dati per l'operando FIFO o Destination si utilizza una struttura definita dall'utente, utilizzare la stessa struttura per entrambi gli operandi.

Struttura CONTROL

Mnemonico	Tipo di dati	Descrizione
.EU	BOOL	Il bit di abilitazione scaricamento indica che l'istruzione FFU è abilitata. Il bit EU viene impostato per prevenire un falso scaricamento quando comincia la scansione del programma.
.DN	BOOL	Il bit di fine è impostato per indicare che il FIFO è pieno (.POS = .LEN).
.EM	BOOL	Il bit di vuoto indica che il FIFO è vuoto. Se .LEN ≤ 0 o .POS < 0, il bit .EM e il bit .DN vengono impostati.
.LEN	DINT	La lunghezza specifica il numero massimo di elementi del FIFO.
.POS	DINT	La posizione identifica il punto fino al quale sono stati caricati dei dati nel FIFO.

Descrizione: Utilizzare l'istruzione FFU con l'istruzione FFL per memorizzare ed estrarre dati nell'ordine primo ad entrare/primo ad uscire.

Se è abilitata, l'istruzione FFU scarica i dati dal primo elemento del FIFO e li inserisce nella destinazione. L'istruzione scarica un valore ogni volta che l'istruzione è abilitata fino a quando il FIFO non è vuoto. Se il FIFO è vuoto, l'FFU invia uno 0 alla destinazione.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione FFU opera su una memoria contigua. In alcuni casi, l'istruzione scarica i dati da altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

Indicatori di stato aritmetico: non influenzati

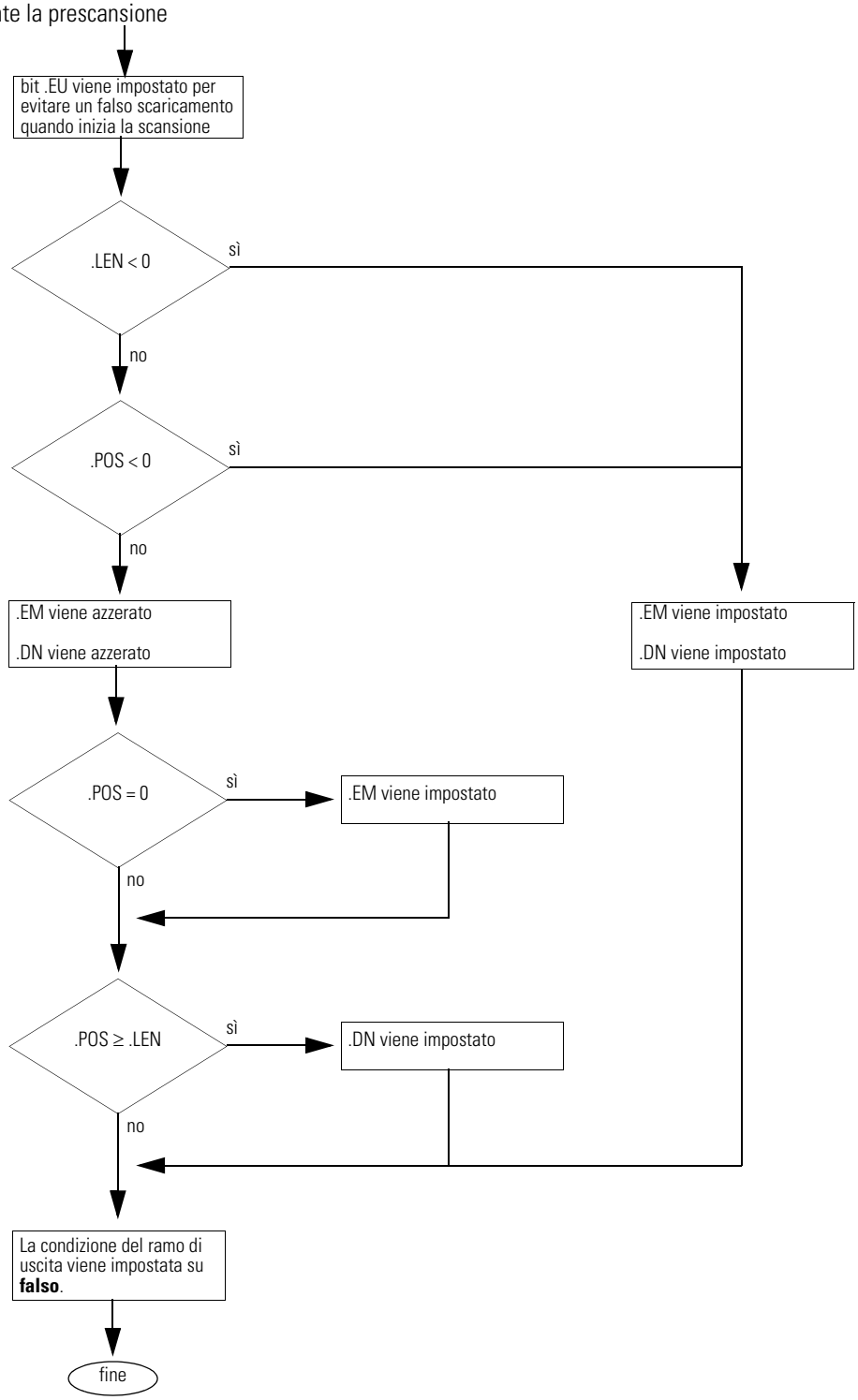
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
Length > dimensione matrice FIFO	4	20

Esecuzione:

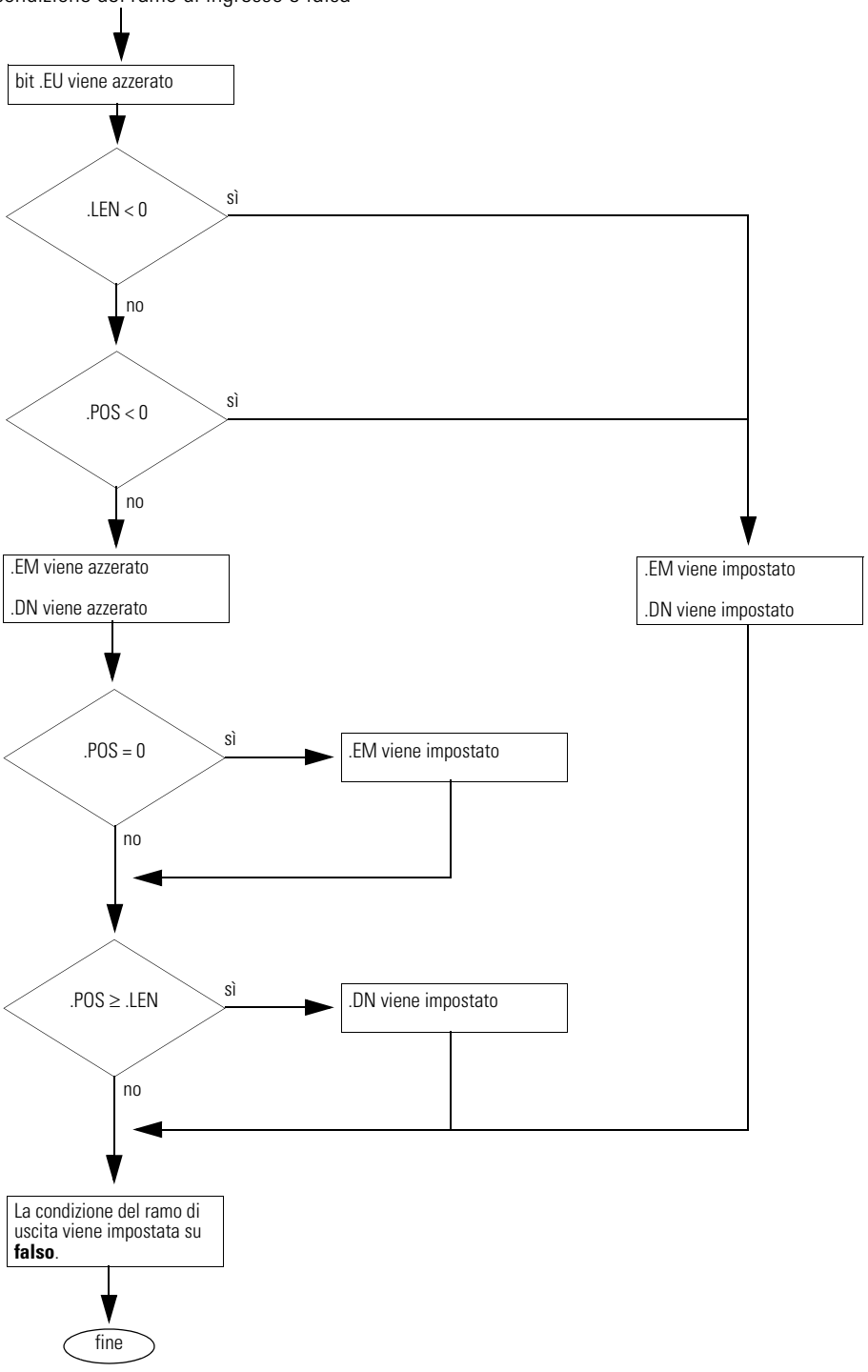
Condizione	Azione ladder

durante la prescansione



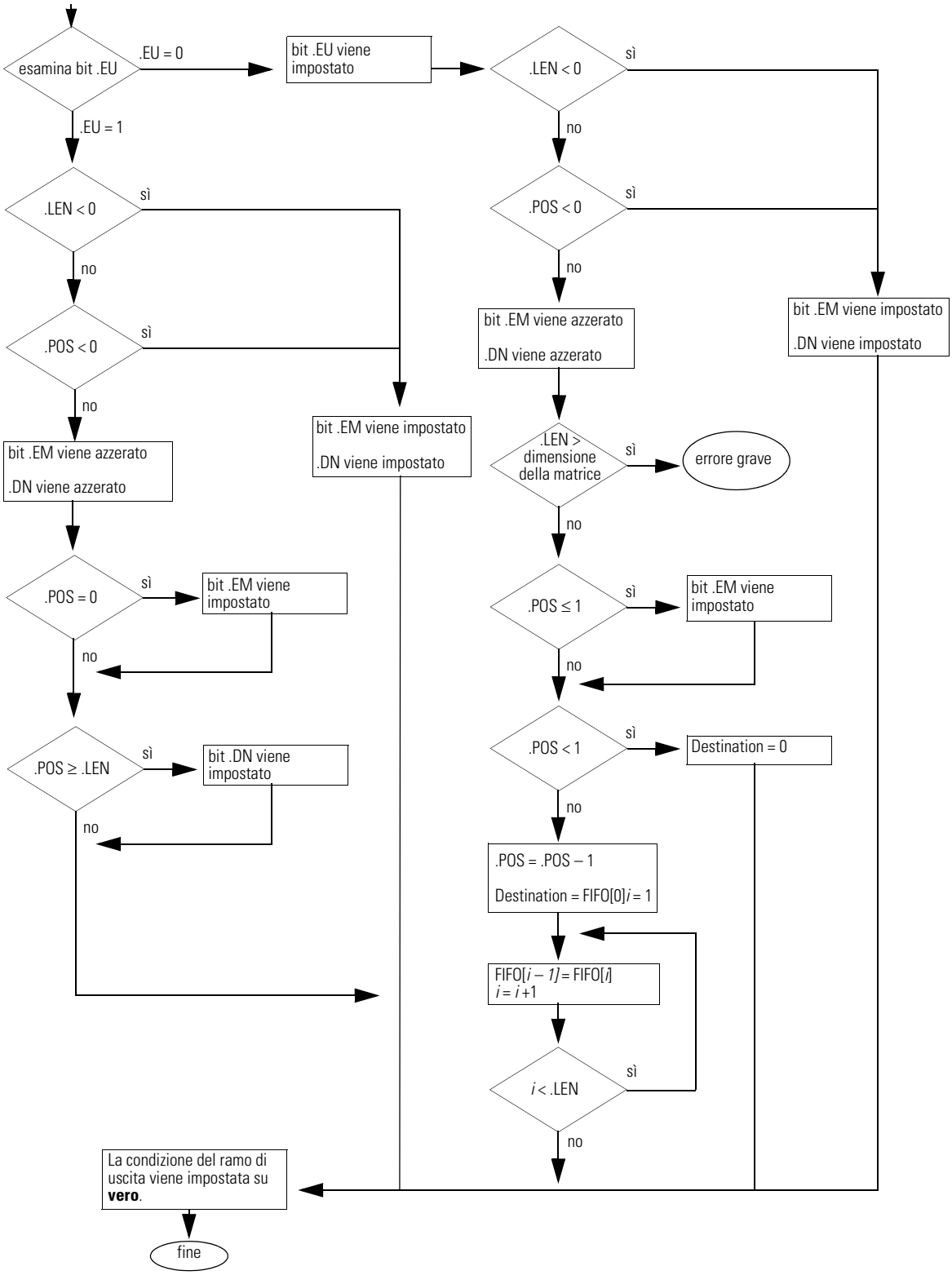
Condizione	Azione ladder

se la condizione del ramo di ingresso è falsa



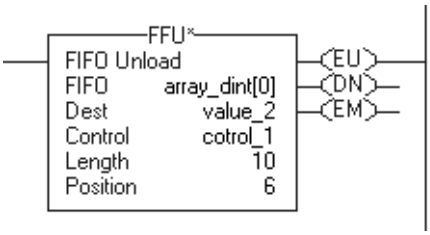
Condizione	Azione ladder

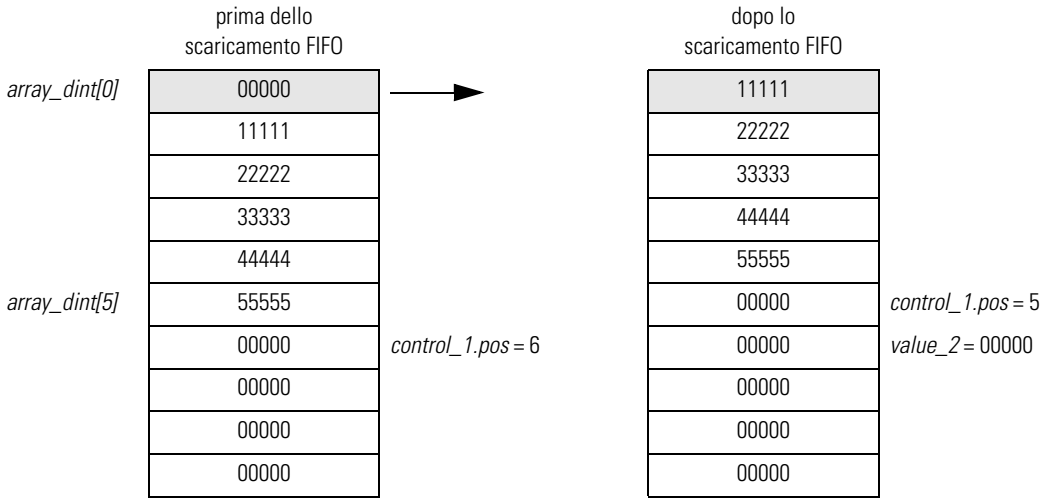
se la condizione del ramo di ingresso è vera



post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione FFU scarica *array_dint[0]* in *value_2* e fa scorrere i rimanenti elementi lungo *array_dint*.



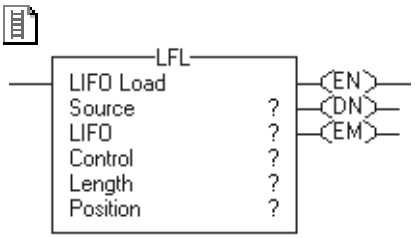


LIFO Load (LFL)

L'istruzione LFL copia il valore di Origine nel LIFO.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	**SINT** **INT** **DINT** **REAL** **stringa** **struttura** Source viene convertito nel tipo di dati del tag matrice. Un intero più piccolo viene convertito in un intero più grande tramite estensione del segno.	immediato tag	dati da memorizzare nel LIFO
LIFO	**SINT** **INT** **DINT** **REAL** **stringa** **struttura**	tag matrice	LIFO da modificare specifica il primo elemento del LIFO **non** usare CONTROL.POS nell'indice
Control	CONTROL	tag	struttura di controllo per l'operazione in genere, utilizzare lo stesso CONTROL del LFU associato
Length	DINT	immediato	numero massimo di elementi che il LIFO può contenere contemporaneamente
Position	DINT	immediato	posizione successiva nel LIFO in cui l'istruzione carica i dati il valore iniziale generalmente è 0

Se per l'operando Source o LIFO si utilizza una struttura definita dall'utente come tipo di dati, utilizzare la stessa struttura per entrambi gli operandi.

Struttura CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione LFL è abilitata.
.DN	BOOL	Il bit di fine è impostato per indicare che il LIFO è pieno (.POS = .LEN). Il bit .DN impedisce il caricamento del LIFO finché .POS < .LEN.
.EM	BOOL	Il bit vuoto indica che il LIFO è vuoto. Se .LEN ≤ 0 o .POS < 0, vengono impostati sia il .EM sia il bit .DN.
.LEN	DINT	La lunghezza specifica il numero massimo di elementi che il LIFO può contenere contemporaneamente.
.POS	DINT	La posizione identifica la posizione del LIFO in cui l'istruzione caricherà il valore successivo.

Descrizione: Utilizzare l'istruzione LFL con l'istruzione LFU per memorizzare ed estrarre dati nell'ordine ultimo ad entrare/primo ad uscire. Se sono usate in coppia, le istruzioni LFL e LFU formano un registro a scorrimento asincrono.

In genere Source e LIFO sono dello stesso tipo di dati.

Se è abilitata, l'istruzione LFL carica il valore dell'origine nella posizione del LIFO identificata dal valore .POS. L'istruzione carica un valore ogni volta che l'istruzione è abilitata fino a quando il LIFO non è pieno.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione LFL opera su una memoria contigua. In alcuni casi, le istruzioni caricano i dati oltre la matrice in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

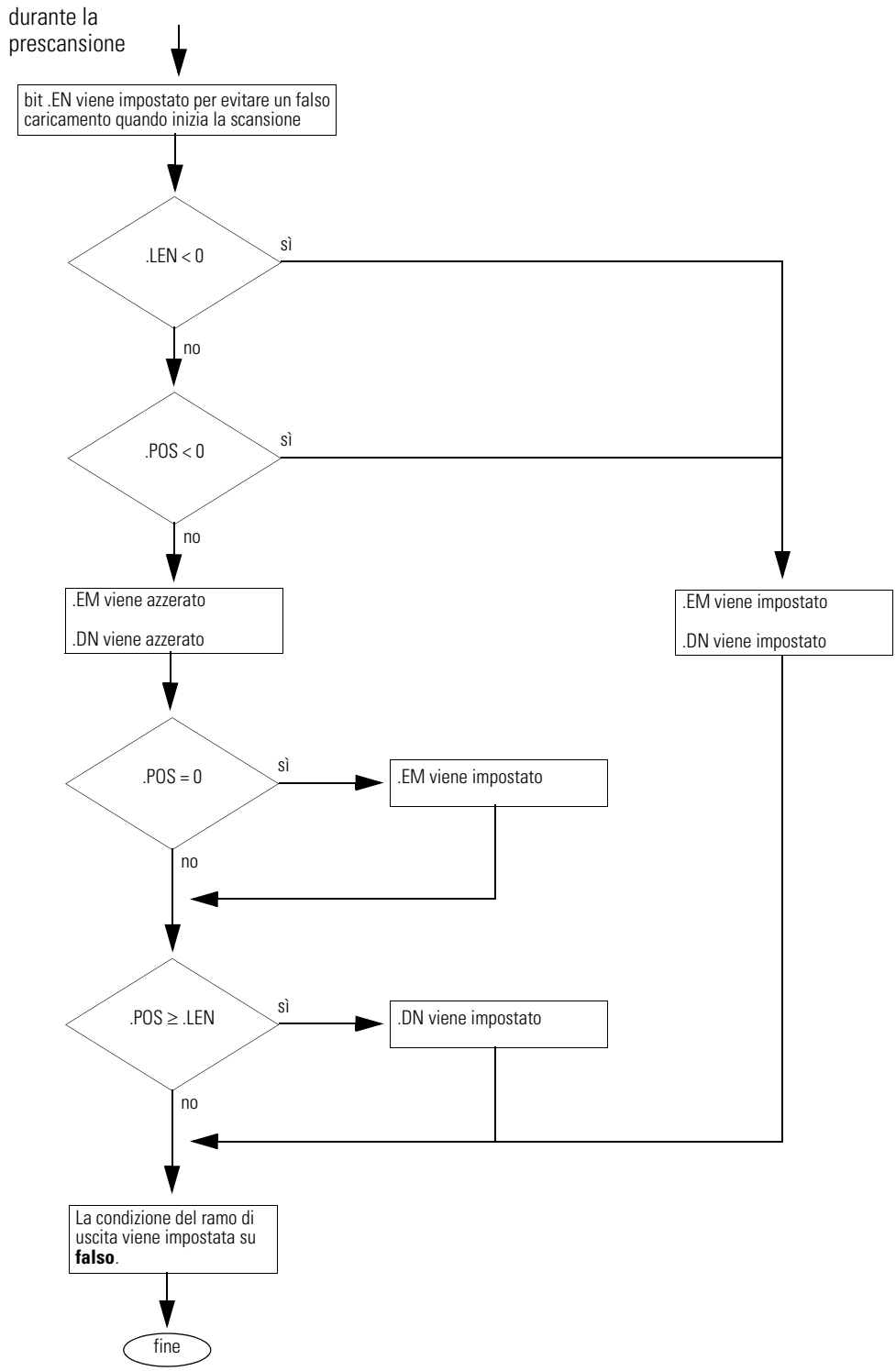
Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

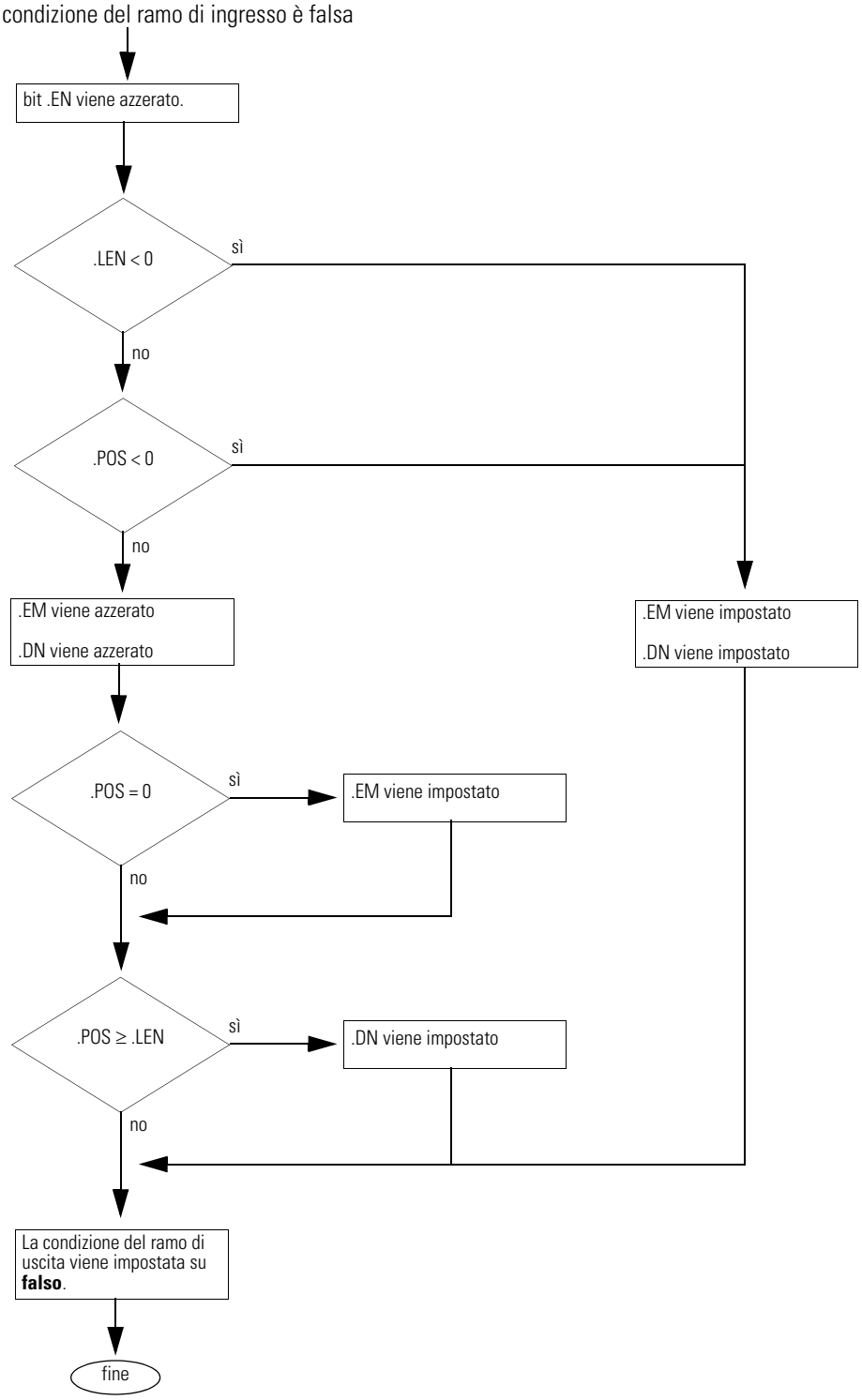
Si verifica un errore grave se	Tipo errore	Codice errore
(elemento iniziale + .POS) > dimensione matrice LIFO	4	20

Esecuzione:

Condizione	Azione ladder

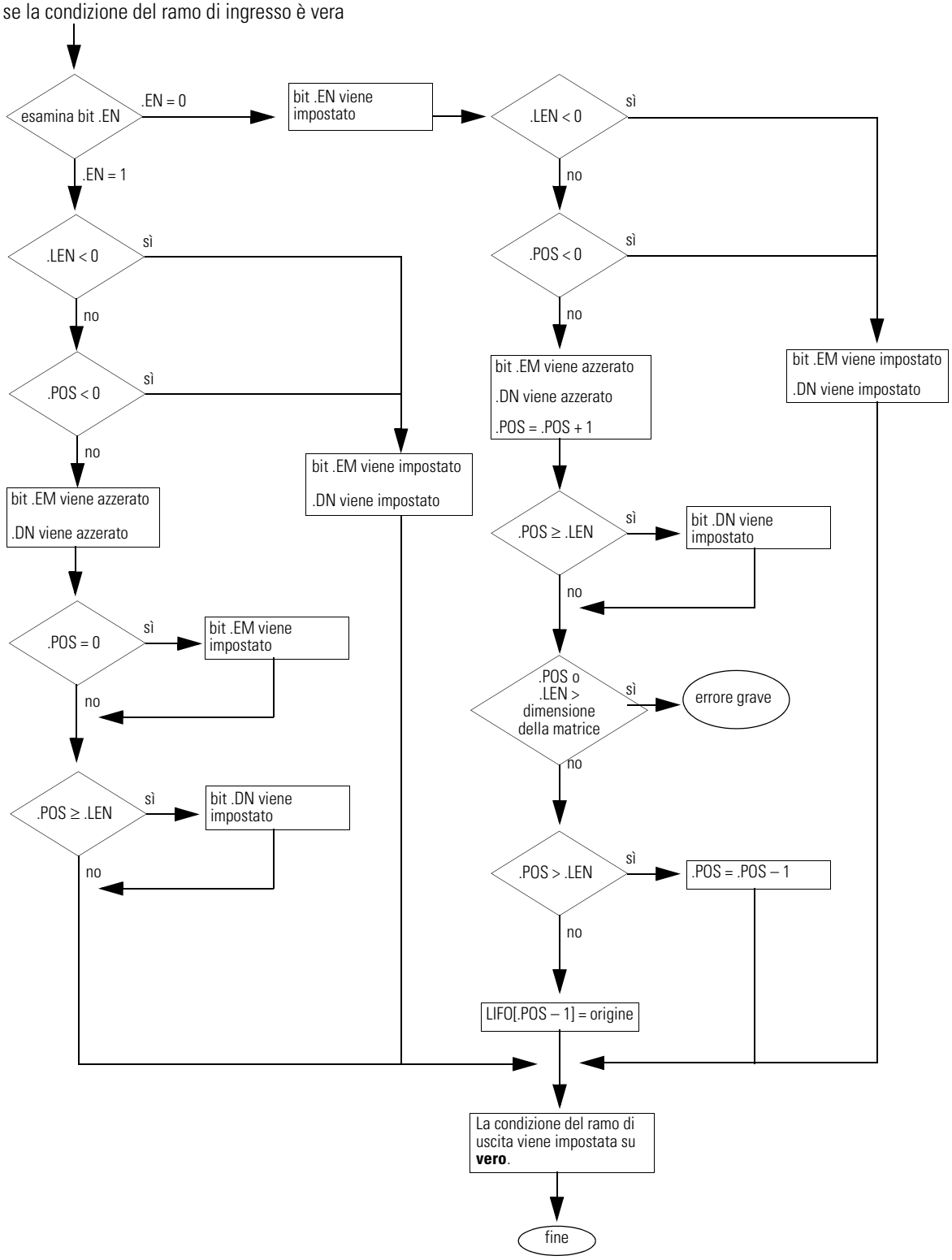


Condizione	Azione ladder
se la condizione del ramo di ingresso è falsa	



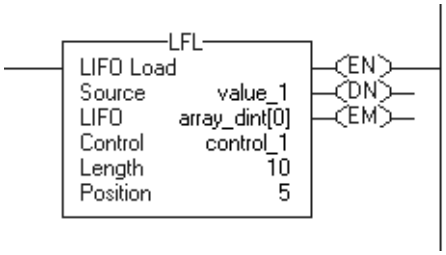
Condizione	Azione ladder

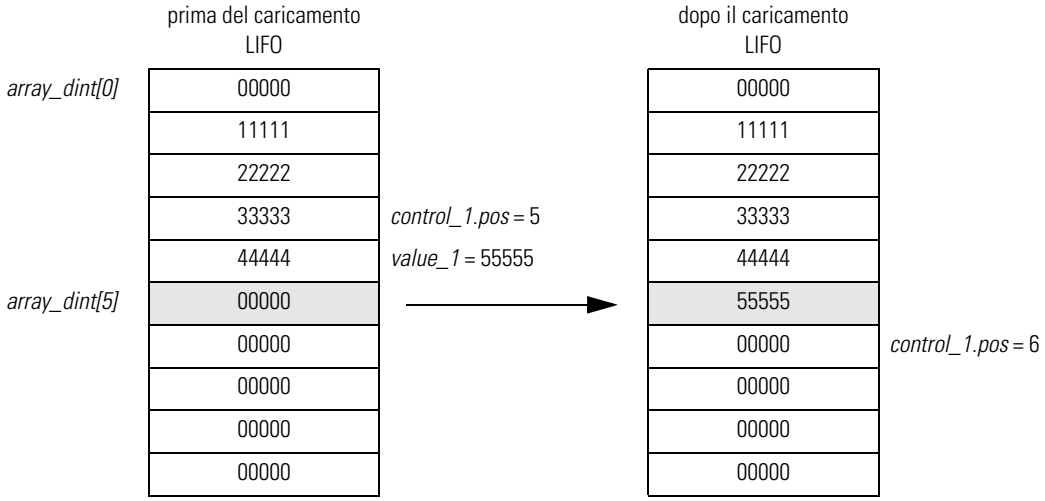
se la condizione del ramo di ingresso è vera



post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione LFL carica *value_1* nella posizione successiva del LIFO che, in questo esempio, è *array_dint[5]*.



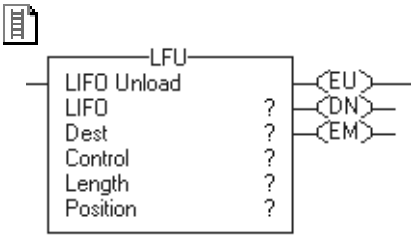


LIFO Unload (LFU)

L'istruzione LFU scarica il valore di. POS del LIFO e memorizza uno 0 in quella posizione.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
LIFO	SINT	tag matrice	LIFO da modificare
	INT		specifica il primo elemento del LIFO
	DINT		**non** usare CONTROL.POS nell'indice
	REAL		
	stringa		
	struttura		
Destination	SINT	tag	valore che esce dal LIFO
	INT		
	DINT		
	REAL		
	stringa		
	struttura		
	Il valore Destination viene convertito nel tipo di dati del tag Destination. Un intero più piccolo viene convertito in un intero più grande tramite estensione del segno.		
Control	CONTROL	tag	struttura di controllo per l'operazione in genere, utilizzare lo stesso CONTROL dell'LFL associato
Length	DINT	immediato	numero massimo di elementi che il LIFO può contenere contemporaneamente
Position	DINT	immediato	posizione successiva nel LIFO in cui l'istruzione scarica i dati il valore iniziale generalmente è 0

Se come tipo di dati per l'operando LIFO o Destination si utilizza una struttura definita dall'utente, utilizzare la stessa struttura per entrambi gli operandi.

Struttura CONTROL

Mnemonico	Tipo di dati	Descrizione
.EU	BOOL	Il bit di abilitazione scaricamento indica che l'istruzione LFU è abilitata. Il bit EU viene impostato per prevenire un falso scaricamento quando comincia la scansione del programma.
.DN	BOOL	Il bit di fine è impostato per indicare che il LIFO è pieno (.POS = .LEN).
.EM	BOOL	Il bit vuoto indica che il LIFO è vuoto. Se .LEN ≤ 0 o .POS < 0, vengono impostati sia il .EM sia il bit .DN.
.LEN	DINT	La lunghezza specifica il numero massimo di elementi che il LIFO può contenere contemporaneamente.
.POS	DINT	La posizione identifica il punto fino al quale sono stati caricati dei dati nel LIFO.

Descrizione: Utilizzare l'istruzione LFU con l'istruzione LFL per memorizzare ed estrarre dati nell'ordine ultimo ad entrare/primo ad uscire.

Se è abilitata, l'istruzione LFU scarica il valore di .POS del LIFO e lo inserisce nella destinazione. L'istruzione scarica un valore e lo sostituisce con uno 0 ogni volta che l'istruzione è abilitata, fino a quando il LIFO non è vuoto. Se il LIFO è vuoto, l'LFU invia uno 0 alla destinazione.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione LFU opera su una memoria contigua. In alcuni casi, l'istruzione scarica i dati da altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

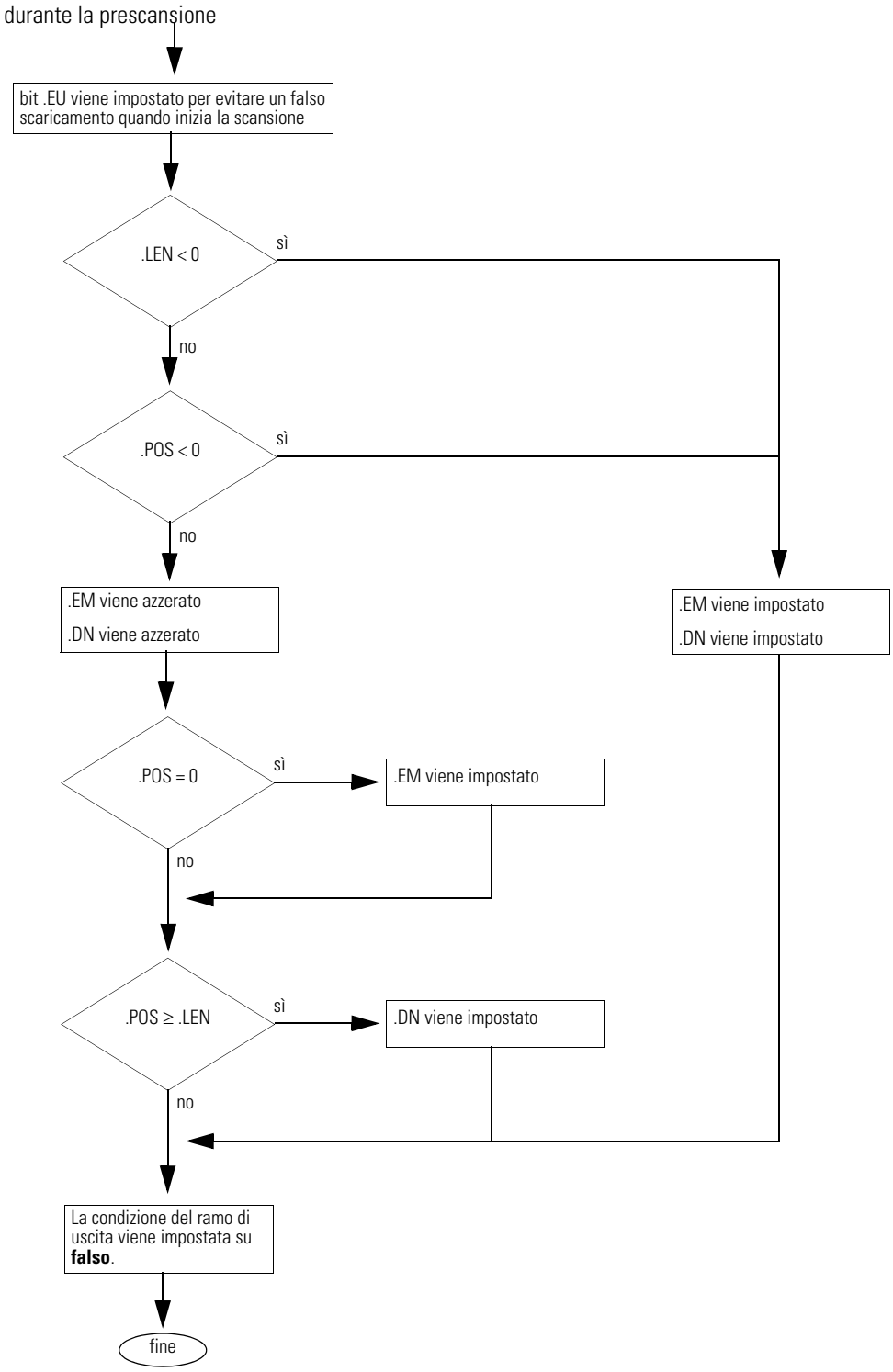
Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

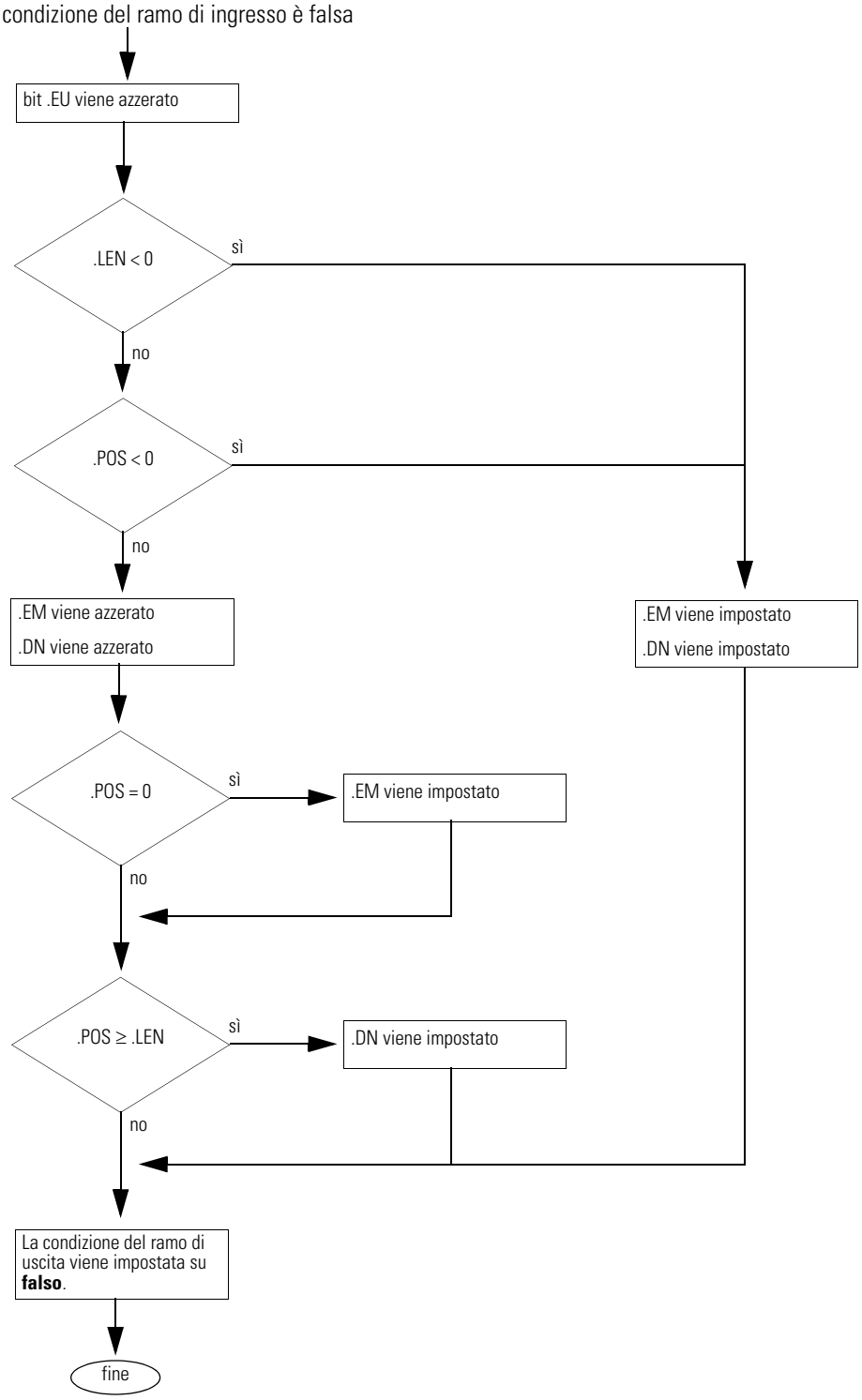
Si verifica un errore grave se	Tipo errore	Codice errore
Length > dimensione matrice LIFO	4	20

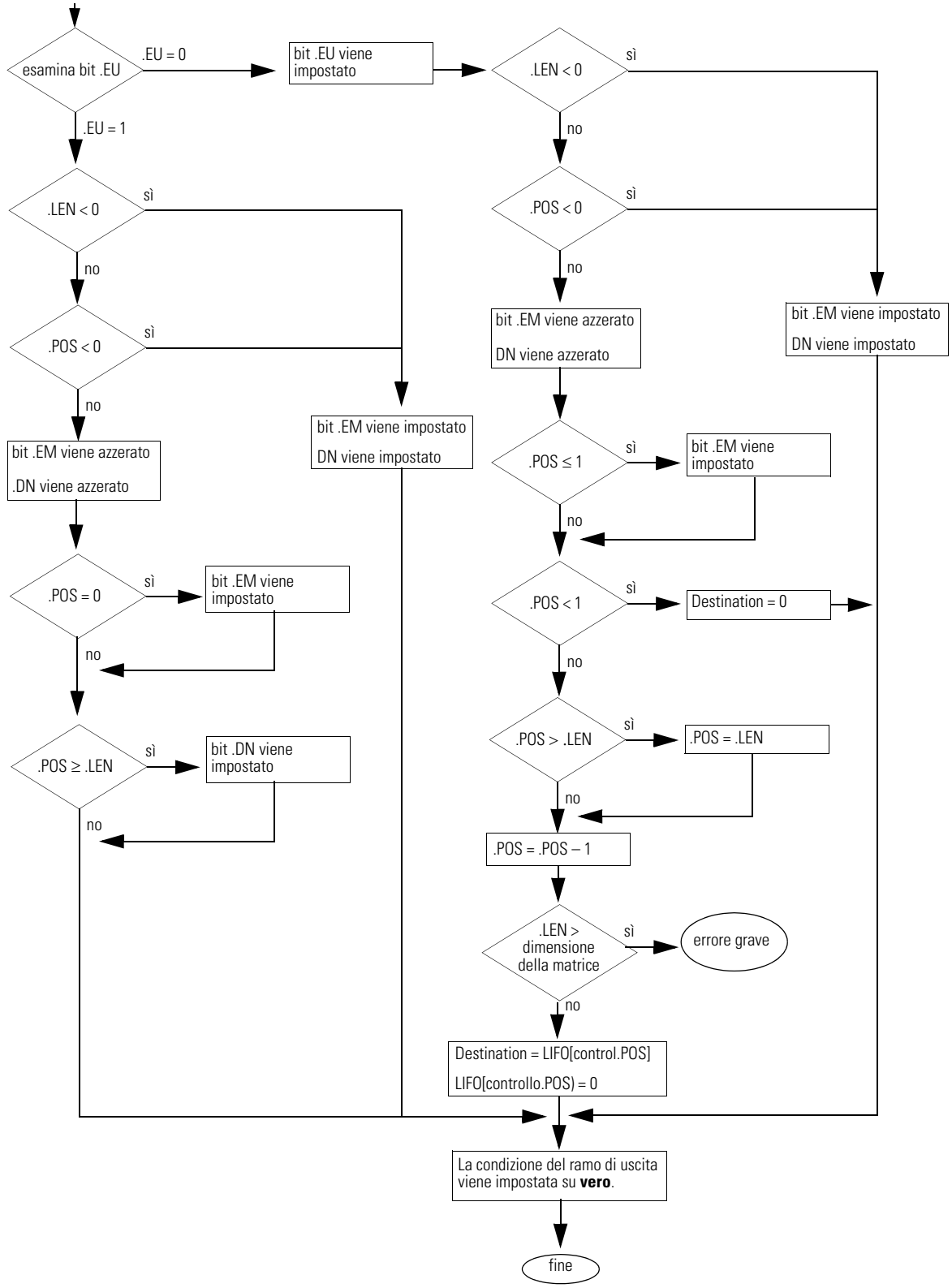
Esecuzione:

Condizione	Azione ladder

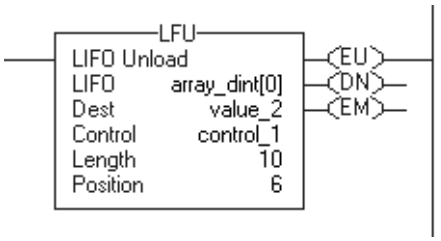


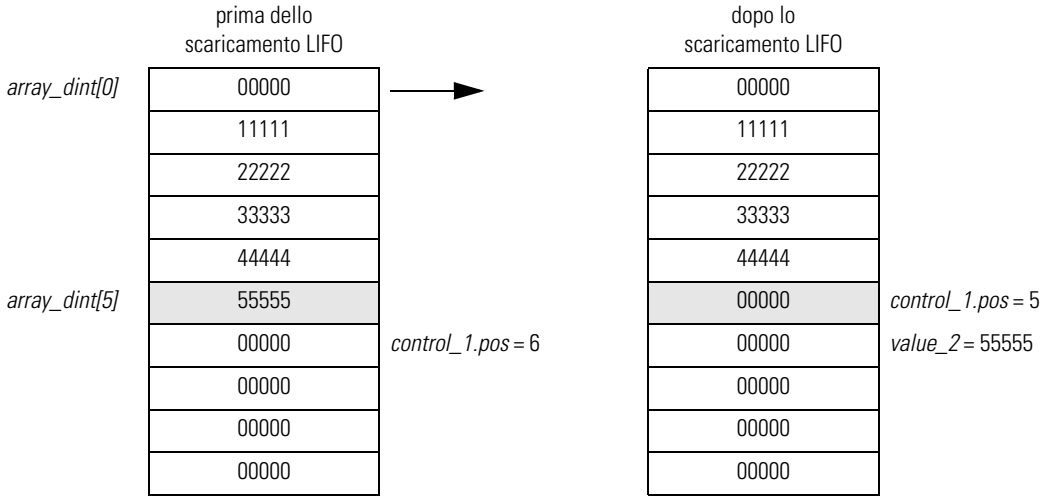
Condizione	Azione ladder
se la condizione del ramo di ingresso è falsa	



Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione LFU scarica *array_dint[5]* in *value_2*.





Note:

Istruzioni sequenziatore
(SQI, SQO, SQL)

Introduzione

Nessuna azione. Le istruzioni sequenziatore controllano operazioni costanti e ripetibili.

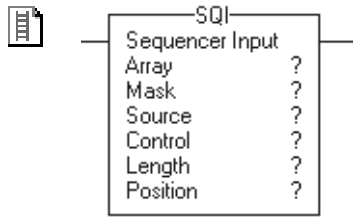
Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
Sapere quando una fase è stata completata.	SQI	ladder	414
Impostare le condizioni di uscita per la fase successiva.	SQO	ladder	418
Caricare le condizioni di riferimento nelle matrici sequenziatore	SQL	ladder	422

Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimali, in genere DINT o REAL.

Sequencer Input (SQI)

L'istruzione SQI rileva quando un passo è completo in una coppia in sequenza di istruzioni SQO/SQI.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Array	DINT	tag matrice	matrice sequenziatore specifica il primo elemento della matrice sequenziatore **non** usare CONTROL.POS nell'indice
Mask	SINT	tag	quali bit bloccare o fare passare
	INT	immediato	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Source	SINT	tag	dati di ingresso della matrice sequenziatore
	INT		
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.		
Control	CONTROL	tag	struttura di controllo per l'operazione generalmente usa lo stesso CONTROL delle istruzioni SQO ed SQL
Length	DINT	immediato	numero di elementi della matrice (tabella sequenziatore) da confrontare
Position	DINT	immediato	posizione corrente nella matrice il valore iniziale generalmente è 0

Struttura di CONTROL

Mnemonico	Tipo di dati	Descrizione
.ER	BOOL	Il bit di errore viene impostato quando .LEN $\leq$ 0, .POS < 0, o .POS > .LEN.
.LEN	DINT	La lunghezza specifica il numero di passi della matrice sequenziatore.
.POS	DINT	La posizione identifica l'elemento che l'istruzione sta confrontando.

Descrizione: Se è abilitata, l'istruzione SQI verifica l'uguaglianza di un elemento Source, tramite una maschera, con un elemento della matrice.

Generalmente usa lo stesso CONTROL delle istruzioni SQO ed SQL.

L'istruzione SQI opera su una memoria contigua.

Immissione di un valore di maschera immediato

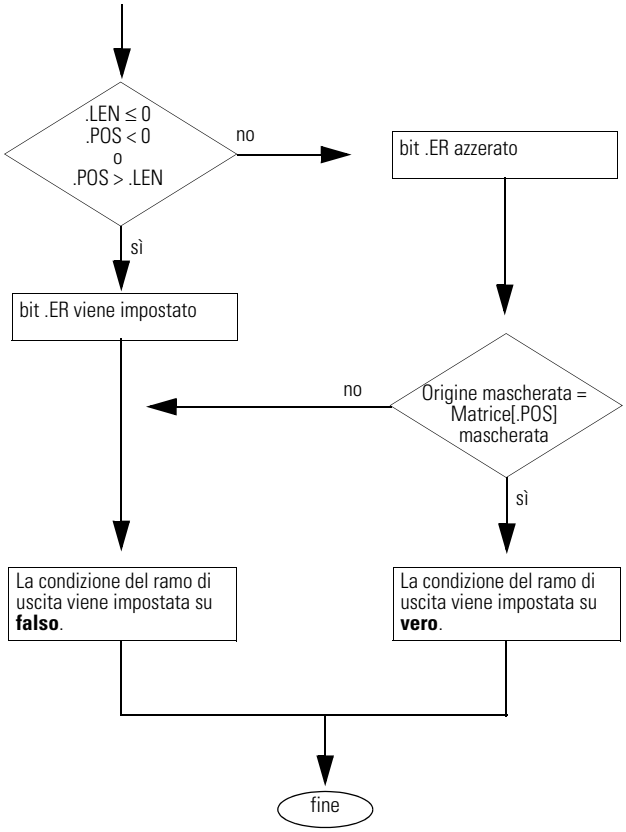
Quando si immette una maschera, il software di programmazione imposta automaticamente valori decimali. Se si desidera immettere una maschera utilizzando un altro formato, fare precedere il prefisso corretto al valore.

Prefisso	Descrizione
16#	Esadecimale per esempio; 16#0F0F
8#	Ottale per esempio; 8#16
2#	Binario per esempio; 2#00110011

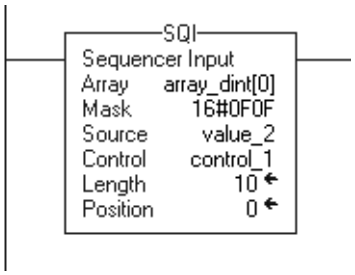
Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione SQI fa passare *value_2* attraverso la maschera per stabilire se il risultato è uguale all'elemento corrente della *array_dint*. Il confronto mascherato è vero per cui la condizione del ramo di uscita diventa vera.



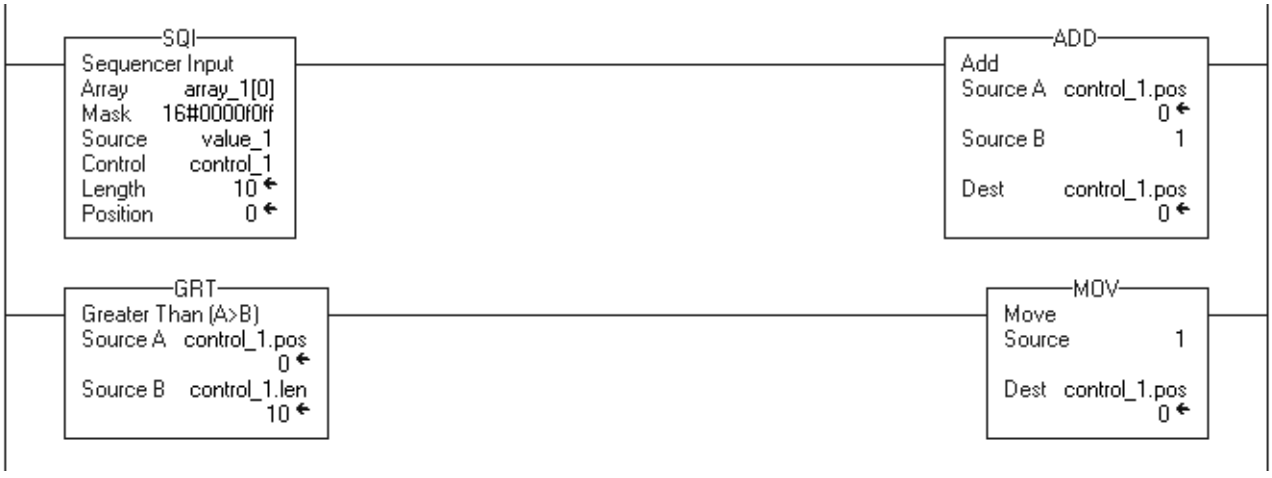
Operando SQI	Valori di esempio (DINT visualizzati in binario)
Source	**xxxxxxxx xxxxxxxx xxxx0101 xxxx1010**
Mask	**00000000 00000000 00001111 00001111**
Array	**xxxxxxxx xxxxxxxx xxxx0101 xxxx1010**

Uno 0 nella maschera significa che il bit non viene confrontato (indicato dalle xxxx in questo esempio).

Uso di SQI senza SQO

Se si utilizza l'istruzione SQI senza un'istruzione SQO, è necessario incrementare dall'esterno la matrice sequenziatore.

L'istruzione SQI confronta il valore di origine. L'istruzione ADD fa incrementare il puntatore della matrice sequenziatore. L'istruzione GRT determina se nella matrice sequenziatore c'è un altro valore da controllare. L'istruzione MOV ripristina il valore della posizione dopo avere completato una volta tutte le fasi della matrice sequenziatore.

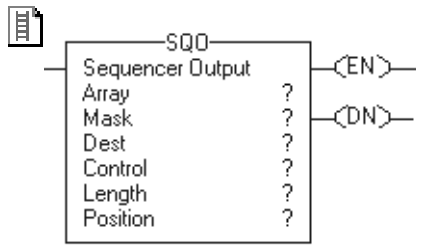


Sequencer Output (SQO)

L'istruzione SQO imposta le condizioni di uscita per il passo successivo di una coppia in sequenza di istruzioni SQO/SQI.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Array	DINT	tag matrice	matrice sequenziatore specifica il primo elemento della matrice sequenziatore **non** usare CONTROL.POS nell'indice
Mask	SINT INT DINT	tag immediato	quali bit bloccare o fare passare Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.
Destination	DINT	tag	dati di uscita della matrice sequenziatore
Control	CONTROL	tag	struttura di controllo per l'operazione generalmente usa lo stesso CONTROL delle istruzioni SQI ed SQL
Length	DINT	immediato	numero di elementi della matrice (tabella sequenziatore) per l'uscita
Position	DINT	immediato	posizione corrente nella matrice il valore iniziale generalmente è 0

Struttura di CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione SQO è abilitata.
.DN	BOOL	Il bit di fine è impostato quando tutti gli elementi specificati sono stati spostati nella destinazione.
.ER	BOOL	Il bit di errore viene impostato quando .LEN ≤ 0, .POS < 0, o .POS > .LEN.
.LEN	DINT	La lunghezza specifica il numero di passi della matrice sequenziatore.
.POS	DINT	La posizione identifica l'elemento che il controllore sta manipolando.

Descrizione:

Se è abilitata, l'istruzione SQO incrementa la posizione, sposta i dati della posizione filtrati dalla maschera e memorizza il risultato nella destinazione. Se .POS > .LEN, l'istruzione ritorna all'inizio della matrice sequenziatore e continua con .POS = 1.

Generalmente usa lo stesso CONTROL delle istruzioni SQI ed SQL.

L'istruzione SQO opera su una memoria contigua.

Immissione di un valore di maschera immediato

Quando si immette una maschera, il software di programmazione imposta automaticamente valori decimali. Se si desidera immettere una maschera utilizzando un altro formato, fare precedere il prefisso corretto al valore.

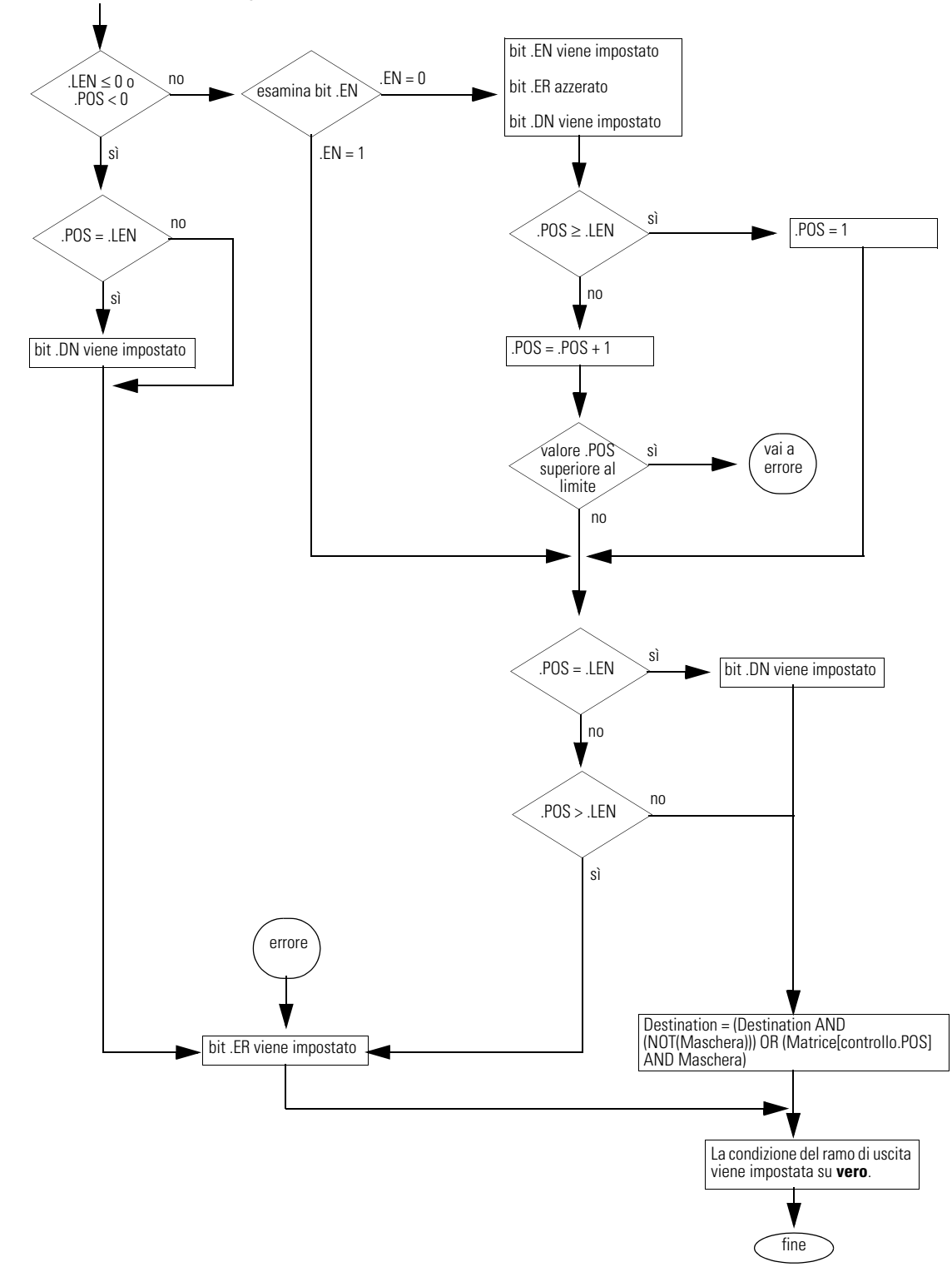
Prefisso	Descrizione
16#	Esadecimale per esempio; 16#0F0F
8#	Ottale per esempio; 8#16
2#	Binario per esempio; 2#00110011

Indicatori di stato aritmetico non influenzati

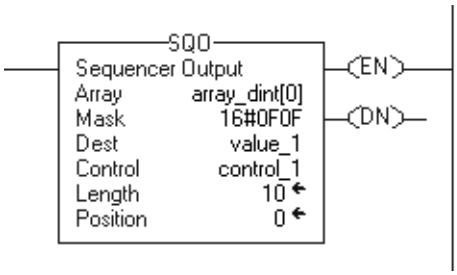
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit .EN viene impostato per prevenire un falso caricamento quando comincia la scansione del programma. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit .EN viene azzerato. La condizione del ramo di uscita viene impostata su falso.

Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione SQO incrementa la posizione, passa i dati di quella posizione in *array_dint* attraverso la maschera e memorizza il risultato in *value_1*.



Operando SQO	Valori di esempio (con INT visualizzati in binario)
Array	**xxxxxxxx xxxxxxxx xxxx0101 xxxx1010**
Mask	**00000000 00000000 00001111 00001111**
Destination	**xxxxxxxx xxxxxxxx xxxx0101 xxxx1010**

Uno 0 nella maschera significa che il bit non viene confrontato (indicato dalle xxxx in questo esempio).

Uso di SQI con SQO

Se si utilizza una coppia di istruzioni SQI ed SQO, assicurarsi che entrambe le istruzioni utilizzino gli stessi valori Control, Length e Position.

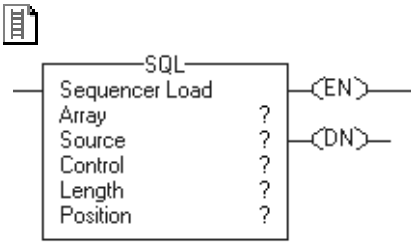
Ripristino della posizione dell'istruzione SQO

Ogni volta che il controllore passa dalla modalità Programmazione alla modalità di Esecuzione, l'istruzione SQO azzera (inizializza) il valore .POS. Per riportare .POS al valore di inizializzazione (.POS = 0), utilizzare un'istruzione RES per azzerare il valore della posizione. In questo esempio si utilizza lo stato del bit di prima scansione per azzerare il valore .POS.

Sequencer Load (SQL)

L'istruzione SQL carica le condizioni di riferimento in una matrice sequenziatore.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Array	DINT	tag matrice	matrice sequenziatore specifica il primo elemento della matrice sequenziatore **non** usare CONTROL.POS nell'indice
Source	SINT INT	tag immediato	dati di ingresso da caricare nella matrice sequenziatore **DINT** Un tag SINT o INT viene convertito in un valore DINT tramite estensione del segno.
Control	CONTROL	tag	struttura di controllo per l'operazione generalmente usa lo stesso CONTROL delle istruzioni SQO ed SQL
Length	DINT	immediato	numero di elementi della matrice (tabella sequenziatore) da caricare
Position	DINT	immediato	posizione corrente nella matrice il valore iniziale generalmente è 0

Struttura di CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione SQL è abilitata.
.DN	BOOL	Il bit di fine viene impostato quando tutti gli elementi specificati sono stati caricati nella matrice.
.ER	BOOL	Il bit di errore viene impostato quando .LEN ≤ 0, .POS < 0, o .POS $>$ .LEN.
.LEN	DINT	La lunghezza specifica il numero di passi della matrice sequenziatore.
.POS	DINT	La posizione identifica l'elemento che il controllore sta manipolando.

Descrizione: Se è abilitata, l'istruzione SQL avanza fino alla posizione successiva della matrice sequenziatore e carica il valore Source in quella posizione. Se il bit .DN è impostato oppure se .POS ≥ .LEN, l'istruzione imposta .POS=1.

Generalmente usa lo stesso CONTROL delle istruzioni SQI ed SQO.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione SQL opera su una memoria contigua. In alcuni casi, le istruzioni caricano i dati oltre la matrice in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

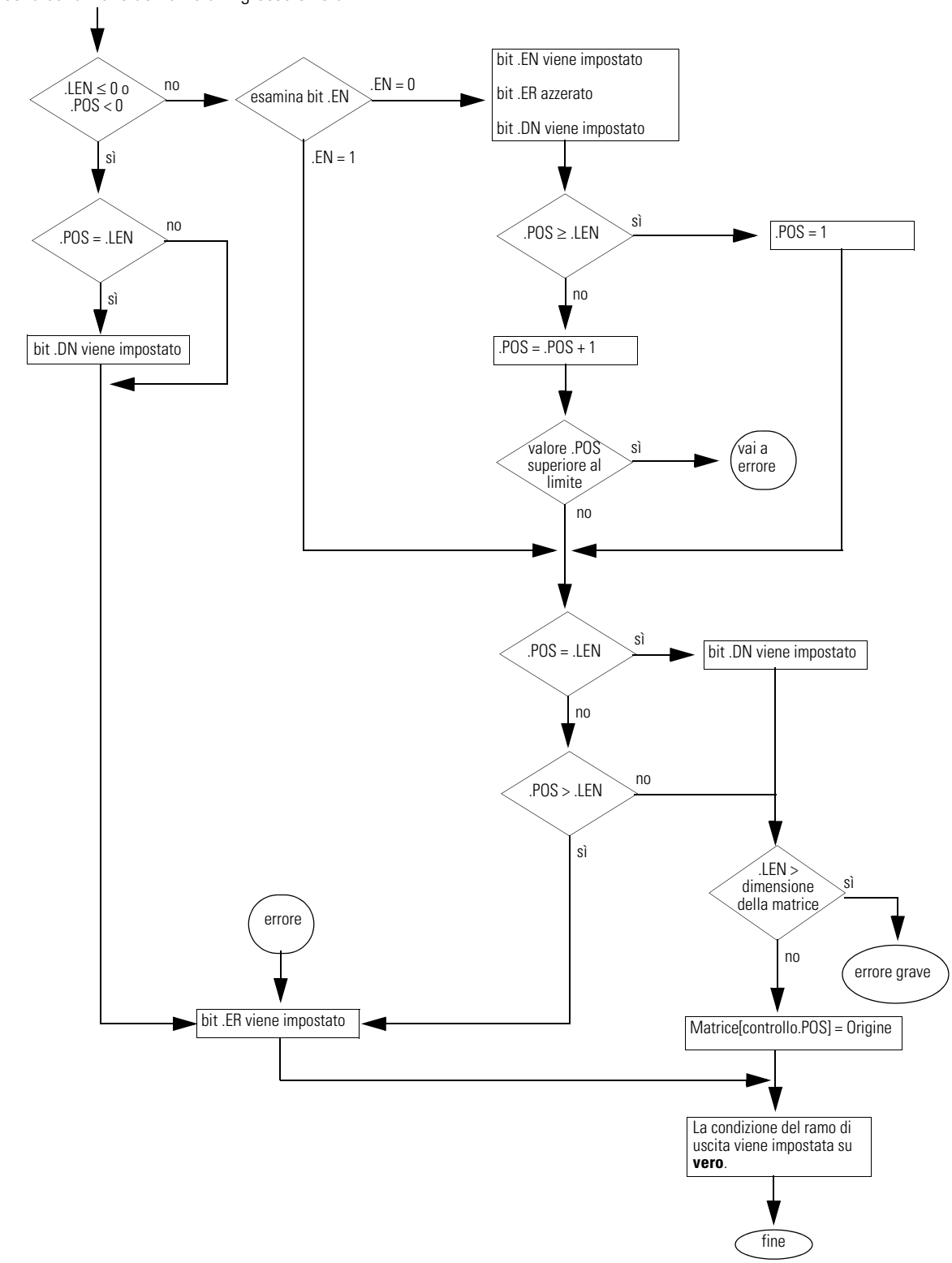
Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

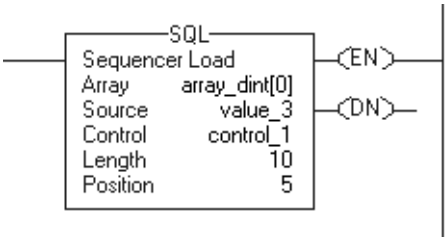
Si verifica un errore grave se	Tipo errore	Codice errore
Length > dimensione della matrice	4	20

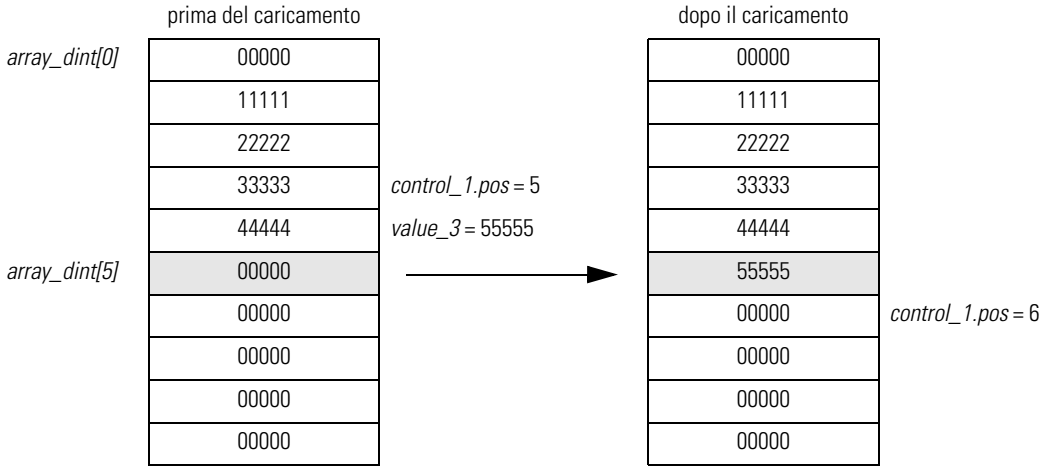
Esecuzione:

Condizione	Azione ladder
durante la prescansione	Il bit .EN viene impostato per prevenire un falso caricamento quando comincia la scansione del programma. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Il bit .EN viene azzerato. La condizione del ramo di uscita viene impostata su falso.

Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione SQL carica il *value_3* nella posizione successiva della matrice sequenziatore, che è la *array_dint[5]* in questo esempio.





Note:

Istruzioni di controllo programma
(JMP, LBL, JSR, RET, SBR, JXR, TND, MCR, UID, UIE, AFI, NOP, EOT, SFP, SFR, EVENT)

Introduzione

Utilizzare le istruzioni di controllo programma per modificare l'esecuzione della logica.

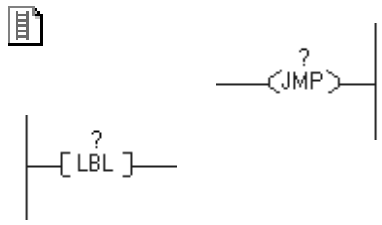
Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
Saltare un segmento di logica che non è necessario eseguire sempre.	JMP LBL	ladder	428
Saltare ad un'altra routine, trasferire dati alla routine, eseguire la routine ed inviare i risultati.	JSR SBR RET	ladder blocco funzione testo strutturato	430
Saltare ad una routine esterna (solo per controllore SoftLogix5800)	JXR	ladder	441
Impostare una fine temporanea che interrompa l'esecuzione della routine.	TND	ladder testo strutturato	444
Disabilitare tutti i rami di una sezione di logica.	MCR	ladder	446
Disabilitare i task utente.	UID	ladder testo strutturato	448
Abilitare i task utente.	UIE	ladder testo strutturato	448
Disabilitare un ramo.	AFI	ladder	450
Inserire un marcatore di posizione nella logica.	NOP	ladder	451
Fine di una transizione per diagramma di funzione sequenziale.	EOT	ladder testo strutturato	452
Pausa di un diagramma di funzione sequenziale.	SFP	ladder testo strutturato	454
Ripristino di un diagramma di funzione sequenziale.	SFR	ladder testo strutturato	456
Attivare l'esecuzione di un task evento.	EVENT	ladder testo strutturato	458

Jump to Label (JMP) Label (LBL)

Le istruzioni JMP ed LBL consentono di saltare porzioni di logica ladder.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Istruzione JMP			
Label name		Nome etichetta	inserire il nome dell'istruzione LBL associata
Istruzione LBL			
Label name		Nome etichetta	il programma salta all'istruzione LBL con il nome di riferimento

Descrizione:

Se è abilitata, l'istruzione JMP va all'istruzione LBL di riferimento ed il controllore continua l'esecuzione da quel punto. Se è disabilitata, l'istruzione JMP non ha alcun effetto sull'esecuzione della logica ladder.

L'istruzione JMP può spostare l'esecuzione ladder in avanti o indietro. Il salto ad una LBL fa diminuire il tempo di scansione del programma poiché si evita un segmento di logica. Il salto indietro fa ripetere al controllore iterazioni di logica.

Fare attenzione a non saltare indietro troppe volte. Il timer watchdog potrebbe scadere perché il controllore non raggiunge mai la fine della logica, e quindi causare l'errore del controllore.

ATTENZIONE

La logica saltata non viene scandita. Inserire la logica critica al di fuori dell'area saltata.

L'istruzione LBL rappresenta la destinazione dell'istruzione JMP con lo stesso nome di etichetta. **Assicurarsi che l'istruzione LBL sia la prima istruzione sul suo ramo.**

Esiste solo un'etichetta con quel nome in una routine. Il nome può:

- contenere un massimo di 40 caratteri
- contenere lettere, numeri e sottolineature (_)

Indicatori di stato aritmetico: non influenzati

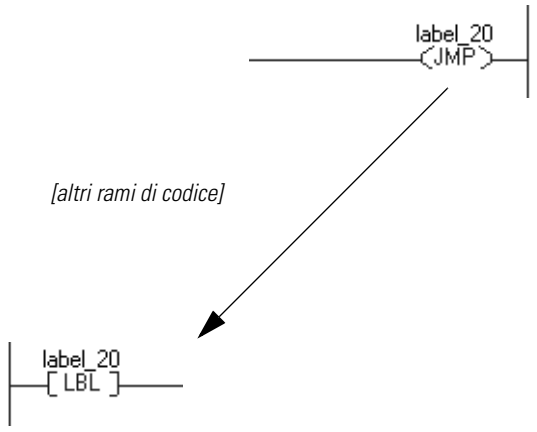
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
l'etichetta non esiste	4	42

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su vero. L'esecuzione salta al ramo che contiene l'istruzione LBL con il nome di etichetta di riferimento.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

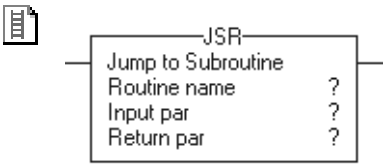
Esempio: Se l'istruzione JMP è abilitata, l'esecuzione salta i rami di logica successivi fino a raggiungere il ramo contenente l'istruzione LBL con *label_20*.



Jump to Subroutine (JSR) Subroutine (SBR) Return (RET)

L'istruzione JSR salta ad una routine diversa. Le istruzioni SBR e RET sono istruzioni opzionali che prevedono lo scambio di dati con l'istruzione JSR.

Operandi JSR:



Ladder

Operando	Tipo	Formato	Descrizione
Routine name	ROUTINE	nome	routine da eseguire (ossia, subroutine)
Input parameter	BOOL SINT INT DINT REAL struttura	immediato tag tag matrice	dati di questa routine che si desidera copiare in un tag della subroutine • I parametri di ingresso sono opzionali. • Se necessario, inserire parametri di ingresso multipli.
Return parameter	BOOL SINT INT DINT REAL struttura	tag tag matrice	tag di questa routine in cui si desidera copiare un risultato della subroutine • I parametri di ritorno sono opzionali. • Se necessario, inserire parametri di ritorno multipli.

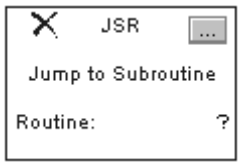
Operandi JSR, continua

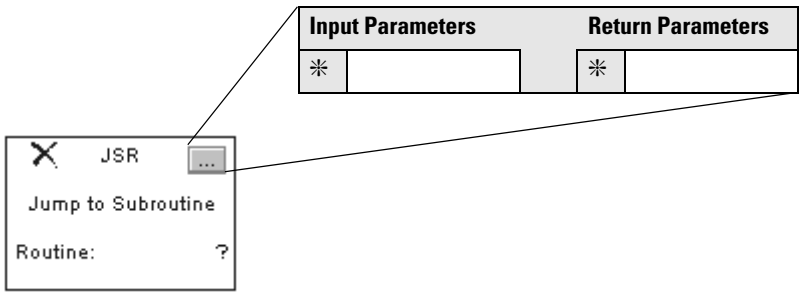
```
JSR(RoutineName,InputCount,
InputPar,ReturnPar);
```

Testo strutturato

Operando	Tipo	Formato	Descrizione
Routine name	ROUTINE	nome	routine da eseguire (ossia, subroutine)
Input count	SINT INT DINT REAL	immediato	numero di parametri di ingresso
Input parameter	BOOL SINT INT DINT REAL struttura	immediato tag tag matrice	dati di questa routine che si desidera copiare in un tag della subroutine • I parametri di ingresso sono opzionali. • Se necessario, inserire parametri di ingresso multipli.
Return parameter	BOOL SINT INT DINT REAL struttura	tag tag matrice	tag di questa routine in cui si desidera copiare un risultato della subroutine • I parametri di ritorno sono opzionali. • Se necessario, inserire parametri di ritorno multipli.

Blocco funzione





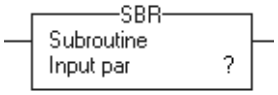
Gli operandi sono gli stessi dell'istruzione JSR in logica ladder.

ATTENZIONE

Per ciascun parametro di un'istruzione SBR o RET, utilizzare lo stesso tipo di dati (incluse le dimensioni della matrice) del corrispondente parametro dell'istruzione JSR. L'utilizzo di tipi di dati diversi può provocare risultati imprevisti.

Operandi SBR: L'istruzione deve essere la prima istruzione in una routine di logica o testo strutturato.

Ladder



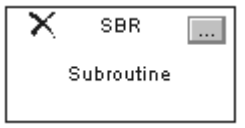
Operando	Tipo	Formato	Descrizione
Input parameter	BOOL	tag	tag di questa routine in cui si desidera copiare il parametro di ingresso corrispondente dall'istruzione JSR
	SINT	tag matrice	
	INT		
	DINT		
	REAL		
	struttura		

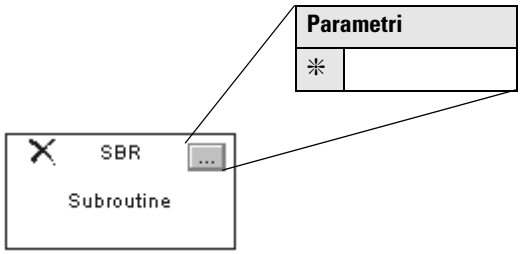
Testo strutturato

```
SBR(InputPar);
```

Gli operandi sono gli stessi dell'istruzione SBR in logica ladder.

Blocco funzione

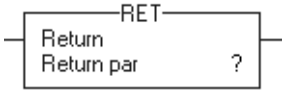




Gli operandi sono gli stessi dell'istruzione SBR in logica ladder.

Operandi RET:

Ladder



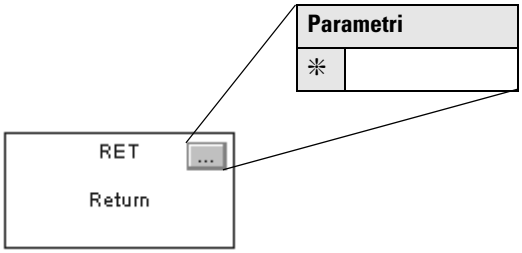
Operando	Tipo	Formato	Descrizione
Return parameter	BOOL	immediato	dati di questa routine che si desidera copiare nel parametro di ritorno corrispondente dell'istruzione JSR
	SINT	tag	
	INT	tag matrice	
	DINT		
	REAL		
	struttura		

Testo strutturato

```
RET(ReturnPar);
```

Gli operandi sono gli stessi dell'istruzione RET in logica ladder.

Blocco funzione



Gli operandi sono gli stessi dell'istruzione RET in logica ladder.

Descrizione: L'istruzione JSR inizia l'esecuzione della routine specificata, definita subroutine:

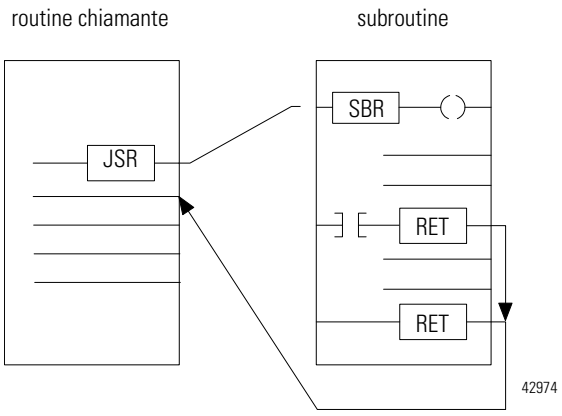
- La subroutine viene eseguita una volta.
- Dopo l'esecuzione della subroutine, l'esecuzione della logica ritorna alla routine che contiene l'istruzione JSR.

Per programmare un salto ad una subroutine, seguire la seguente procedura:

IMPORTANTE Non utilizzare un'istruzione JSR per richiamare (eseguire) la routine principale.

- Si può inserire un'istruzione JSR nella routine principale o in qualunque altra routine.
- Se si utilizza un'istruzione JSR per richiamare la routine principale e quindi si inserisce un'istruzione RET nella routine principale, si verifica un errore grave (tipo 4, codice 31).

Nello schema che segue è illustrato il funzionamento delle istruzioni.



JSR

1. Se si desidera copiare i dati in un tag della subroutine, inserire un parametro di ingresso.
2. Se si desidera copiare un risultato della subroutine in un tag di questa routine, inserire un parametro di ritorno.
3. Inserire il numero di parametri di ingresso e di ritorno necessari.

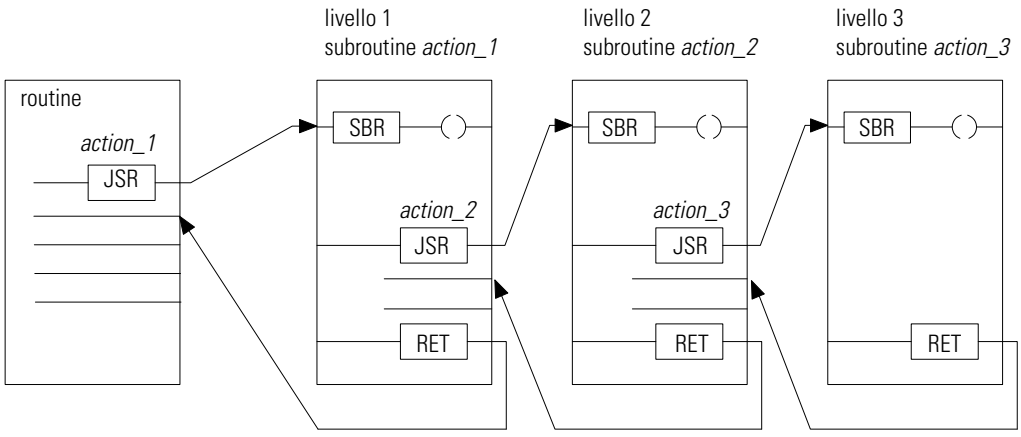
SBR

1. Se l'istruzione JSR ha un parametro di ingresso, inserire un'istruzione SBR.
2. Posizionare l'istruzione SBR come prima istruzione della routine.
3. Per ciascun parametro di ingresso dell'istruzione JSR inserire il tag in cui si desidera copiare i dati.

RET

1. Se l'istruzione JSR ha un parametro di ritorno, inserire un'istruzione RET.
2. Posizionare l'istruzione RET come ultima istruzione della routine.
3. Per ciascun parametro di ritorno dell'istruzione JSR inserire un parametro di ritorno da inviare all'istruzione JSR.
4. In una routine in ladder, se necessario, inserire delle istruzioni RET aggiuntive per uscire dalla subroutine in condizioni di ingresso diverse. (Le routine di blocco funzione consentono di utilizzare una sola istruzione RET.)

Il numero di routine annidate che è possibile avere o di parametri che è possibile passare o ritornare dipende solo dalla memoria del controllore.



Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

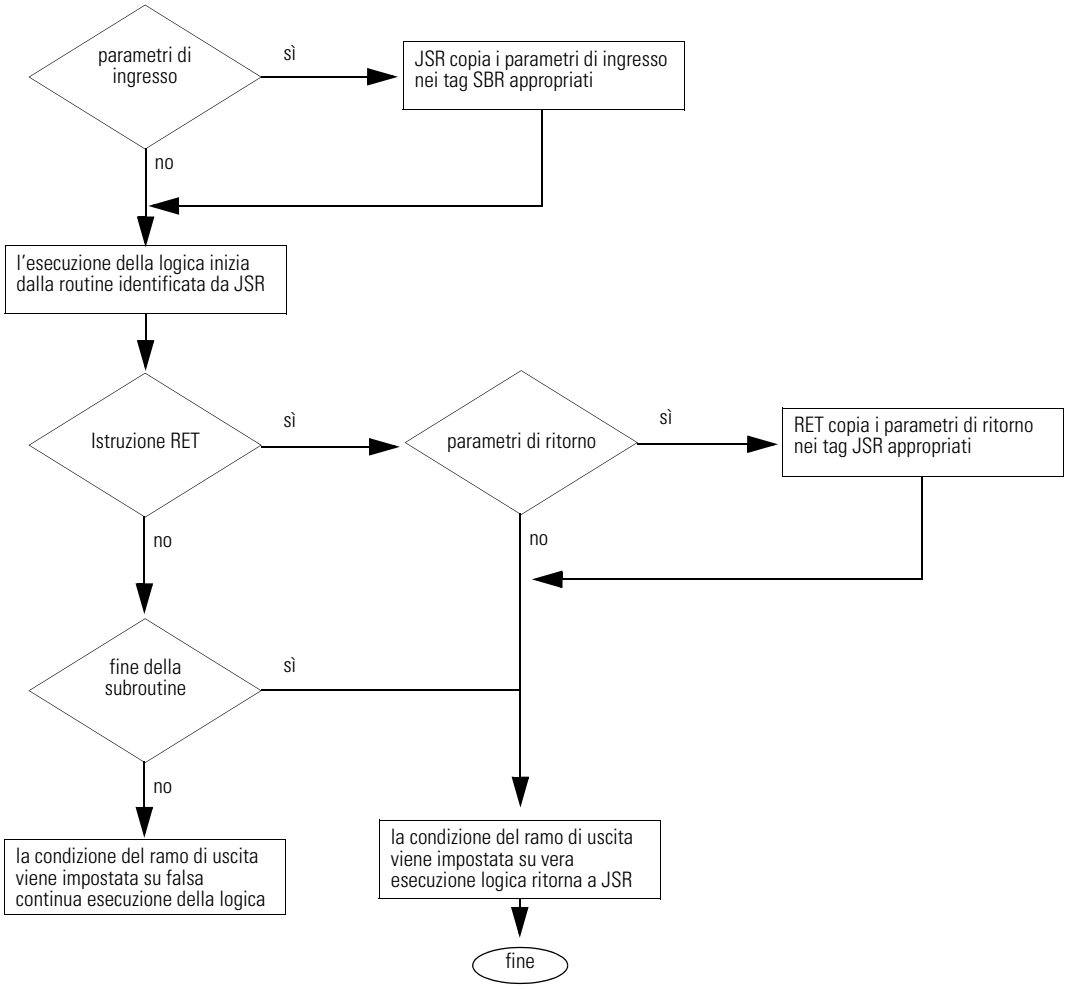
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
l'istruzione JSR ha meno parametri di ingresso dell'istruzione SBR	4	31
l'istruzione JSR salta ad una routine di errore	4 o fornito dall'utente	0 o fornito dall'utente
l'istruzione RET ha meno parametri di ritorno dell'istruzione JSR	4	31
la routine principale contiene un'istruzione RET	4	31

Esecuzione:

Ladder e testo strutturato

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	Il controllore esegue tutte le subroutine, a prescindere dalla condizione del ramo. Per assicurarsi che tutti i rami nella subroutine siano prescansionati, il controllore ignora le istruzioni RET (cioè, le istruzioni RET non chiudono la subroutine). • Con la versione 6.x e precedenti, i parametri di input e return sono accettati. • Con la versione 7.x e successive, i parametri di input e return non sono accettati. Le subroutine ricorsive, vengono prescansionate solo la prima volta. Una subroutine richiamata più volte (non ricorsiva), viene prescansionata ogni volta. La condizione del ramo di uscita viene impostata su falso (solo in ladder).	
l'istruzione del ramo di ingresso dell'istruzione JSR è falsa	La subroutine *non* viene eseguita. Le uscite della subroutine rimangono nel loro ultimo stato. La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.

Condizione	Azione ladder	Azione testo strutturato
esecuzione istruzioni		
post-scansione	Stessa azione della prescansione sopra descritta.	Stessa azione della prescansione sopra descritta.

Blocco funzione

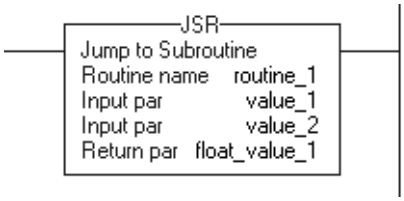
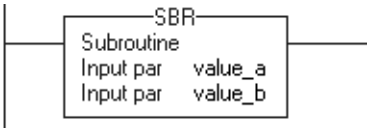
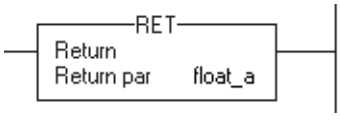
Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
esecuzione normale	1. Se la routine contiene un'istruzione SBR il controllore esegue l'istruzione SBR per prima. 2. Il controllore aggancia tutti i valori dei dati sugli IREF. 3. Il controllore esegue gli altri blocchi funzione nell'ordine determinato dai relativi cablaggi. Questo implica l'esecuzione di altre istruzioni JSR. 4. Il controllore scrive le uscite negli OREF. 5. Se la routine contiene un'istruzione RET il controllore esegue l'istruzione RET per ultima.
post-scansione	Chiamata alla subroutine. Se la routine è di tipo SFC viene inizializzata con le stesse modalità della prescansione.

Esempio 1: L'istruzione JSR passa *value_1* e *value_2* alla *routine_1*.

L'istruzione SBR riceve *value_1* e *value_2* dall'istruzione JSR e copia questi valori in *value_a* e *value_b*, rispettivamente. L'esecuzione della logica continua in questa routine.

L'istruzione RET invia *float_a* all'istruzione JSR. L'istruzione JSR riceve *float_a* e copia il valore in *float_value_1*. L'esecuzione della logica continua con l'istruzione successiva all'istruzione JSR.

Ladder

Routine	Programma
routine principale	
subroutine	 *[altri rami di codice]* 

Testo strutturato

Routine	Programma
routine principale	`JSR(routine_1,2,value_1,value_2,float_value_1);`

subroutine	`SBR(value_a,value_b);`
	`<istruzioni>;`
	`RET(float_a);`

Esempio 2:

Ladder

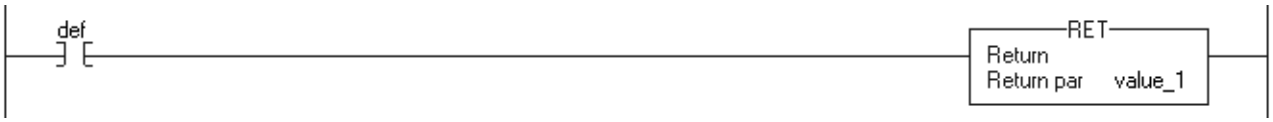
MainRoutine

Se *abc* è attivo, la *subroutine_1* viene eseguita, viene calcolato il numero di biscotti ed il valore viene inserito in *cookies_1*.

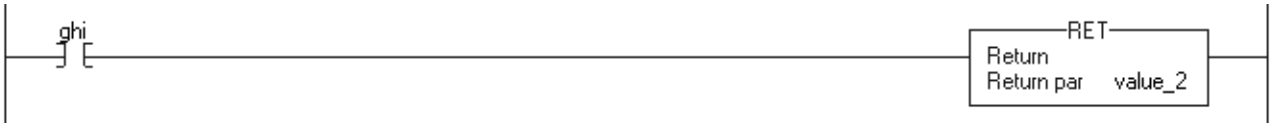
Somma il valore di *cookies_1* con quello di *cookies_2* e memorizza il risultato in *total_cookies*.

Subroutine_1

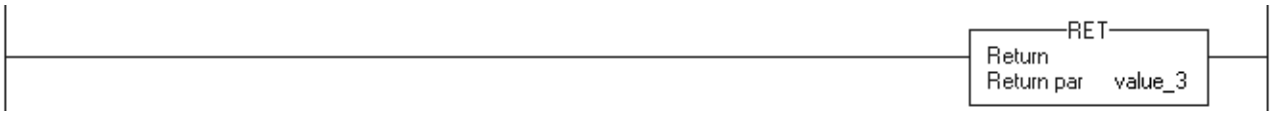
Se *def* è attivo, l'istruzione RET ritorna *value_1* al parametro *cookies_1* di JSR ed il resto della subroutine non viene scandito.



Se *def* è disabilitato (ramo precedente) e *ghi* è abilitato, l'istruzione RET ritorna *value_2* al parametro *cookies_1* di JSR ed il resto della subroutine non viene scandito.

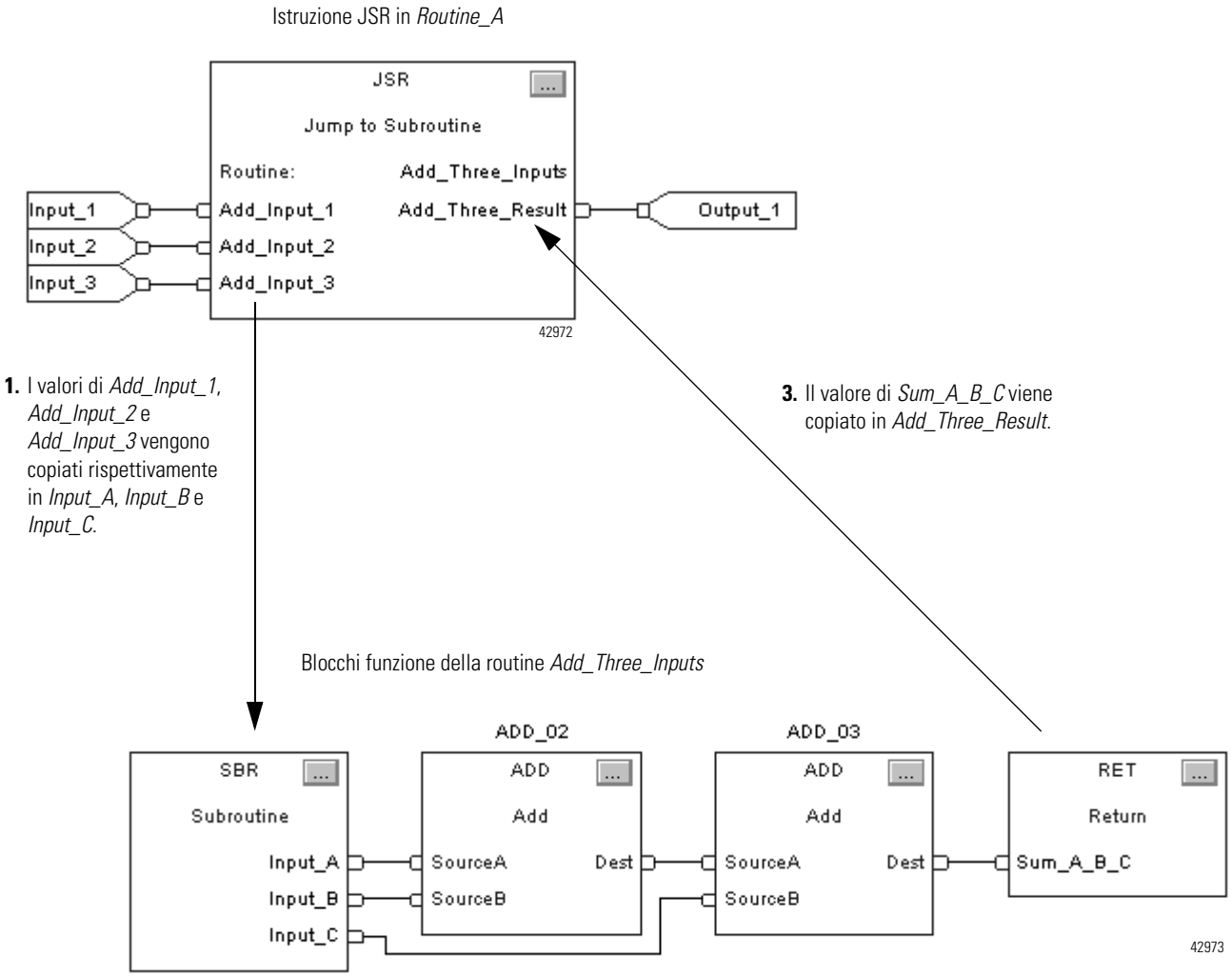


Se sia *def* sia *ghi* sono disabilitati (rami precedenti), l'istruzione RET ritorna *value_3* al parametro *cookies_1* di JSR.



Esempio 3:

Blocco funzione



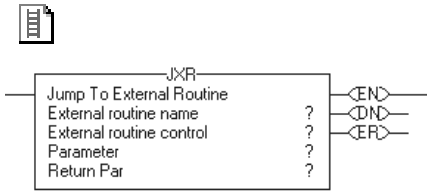
2. L'istruzione ADD somma *Input_A*, *Input_B* e *Input_C* e inserisce il risultato in *Sum_A_B_C*.

Jump to External Routine (JXR)

L'istruzione JXR esegue una routine esterna. Questa istruzione è supportata soltanto dai controllori SoftLogix5800.

Operandi:

Ladder



.

Operando	Tipo	Formato	Descrizione
External routine name	ROUTINE	nome	routine esterna da eseguire
External routine control	EXT_ROUTINE_ CONTROL	tag	struttura di controllo (v. pagina seguente)
Parameter	BOOL SINT INT DINT REAL struttura	immediato tag tag matrice	dati di questa routine che si desidera copiare in una variabile della routine esterna • I parametri sono opzionali. • Se necessario, inserire parametri multipli. • È possibile inserire un massimo di 10 parametri.
Return parameter	BOOL SINT INT DINT REAL	tag	tag di questa routine in cui si desidera copiare un risultato della subroutine esterna • Il parametro di ritorno è opzionale. • È possibile inserire un solo parametro di ritorno.

Struttura EXT_ROUTINE_CONTROL

Mnemonico	Tipo di dati	Descrizione	Implementazione
ErrorCode	SINT	Se si verifica un errore, questo valore corrisponde all'errore. Valori validi nell'intervallo 0–255.	Non esistono codici di errore predefiniti. Lo sviluppatore della routine esterna deve definire i codici di errore.
NumParams	SINT	Questo valore indica il numero di parametri associati a questa istruzione.	Sola visualizzazione – questa informazione viene ricavata da quanto inserito nell'istruzione.
ParameterDefs	EXT_ROUTINE_ PARAMETERS[10]	Questa matrice contiene le definizioni dei parametri da passare alla routine esterna. L'istruzione consente di passare fino a 10 parametri.	Sola visualizzazione – questa informazione viene ricavata da quanto inserito nell'istruzione.
ReturnParamDef	EXT_ROUTIN_ PARAMETERS	Questo valore contiene le definizioni del parametro di ritorno proveniente dalla routine esterna. È previsto un solo parametro di ritorno.	Sola visualizzazione – questa informazione viene ricavata da quanto inserito nell'istruzione.
EN	BOOL	Se impostato, il bit di abilitazione indica che l'istruzione JXR è abilitata.	Questo bit viene impostato dalla routine esterna.
ReturnsValue	BOOL	Se impostato, questo bit indica che è stato inserito un parametro di ritorno per l'istruzione. Se azzerato, questo bit indica che non è stato inserito alcun parametro di ritorno per l'istruzione.	Sola visualizzazione – questa informazione viene ricavata da quanto inserito nell'istruzione.
DN	BOOL	Il bit di fine viene impostato quando la routine esterna è stata eseguita una volta fino alla fine.	Questo bit viene impostato dalla routine esterna.
ER	BOOL	Questo bit di errore viene impostato se si verifica un errore. L'istruzione interrompe la sua esecuzione fino a quando il programma non azzera il bit di errore.	Questo bit viene impostato dalla routine esterna.
FirstScan	BOOL	Questo bit indica se si tratta della prima scansione dopo aver portato il controllore in modalità Run. Se necessario, FirstScan può essere utilizzato per inizializzare la routine esterna.	Questo bit viene impostato dal controllore per riflettere lo stato della scansione.
EnableOut	BOOL	Abilita l'uscita.	Questo bit viene impostato dalla routine esterna.
EnableIn	BOOL	Abilita gli ingressi.	Questo bit viene impostato dal controllore per riflettere la condizione del ramo di ingresso. L'istruzione viene eseguita indipendentemente dalle condizioni del ramo. Lo sviluppatore della routine esterna dovrà monitorare questo stato e agire di conseguenza.
User1	BOOL	Questi bit sono a disposizione dell'utente. Essi non vengono inizializzati dal controllore.	Questi bit possono essere impostati dalla routine esterna o dal programma utente.
User0	BOOL		

Mnemonico	Tipo di dati	Descrizione	Implementazione
ScanType1	BOOL	Questi bit indicano il tipo di scansione della scansione corrente:	Questi bit vengono impostati dal controllore per indicare lo stato della scansione.
ScanType0	BOOL		

Valori bit:	Tipo di scansione:
00	Normale
01	Prescansione
10	Post-scansione (non applicabile ai programmi in ladder)

Descrizione: L'istruzione Salta alla routine esterna (JXR) serve per richiamare la routine esterna da una routine ladder del progetto in uso. L'istruzione JXR supporta vari parametri, per cui è possibile passare i valori dalla routine ladder alla routine esterna e viceversa.

L'istruzione JXR è simile all'istruzione Salta alla subroutine (JSR). L'istruzione JXR inizia l'esecuzione della routine esterna specificata:

- La routine esterna viene eseguita una volta.
- Dopo l'esecuzione della routine esterna, l'esecuzione della logica ritorna alla routine che contiene l'istruzione JXR.

Indicatori di stato aritmetico: Gli indicatori di stato aritmetici non sono influenzati.

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
• si verifica un'eccezione nella DLL della routine esterna • impossibile caricare la DLL • impossibile trovare il punto d'ingresso della DLL	4	88

Esecuzione: La JXR può essere sincrona o asincrona a seconda dell'implementazione della DLL. Il codice della DLL determina anche la modalità di risposta allo stato della scansione, stato condizione del ramo di ingresso e stato condizione del ramo di uscita.

Per ulteriori informazioni sull'uso dell'istruzione JXR e sulla creazione di routine esterne si rimanda a: Manuale utente del sistema SoftLogix5800, pubblicazione 1789-UM002.

Temporary End (TND)

L'istruzione TND agisce come una delimitazione.

Operandi:

Operandi ladder

nessuno

```
TND();
```

Testo strutturato

nessuno

È necessario inserire le parentesi ( ) dopo lo mnemonico dell'istruzione, anche se non ci sono operandi.

Descrizione: Se è abilitata, l'istruzione TND permette al controllore di eseguire la logica solo fino a questa istruzione.

Se è abilitata, l'istruzione TND agisce come fine della routine. Quando il controllore esegue la scansione di un'istruzione TND, il controllore si sposta alla fine della routine corrente. Se l'istruzione TND si trova in una subroutine, il controllo ritorna alla routine chiamante. Se l'istruzione TND si trova invece in una routine principale, il controllo ritorna al programma successivo del task corrente.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

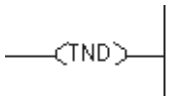
Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	La routine corrente termina.	La routine corrente termina.
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: È possibile utilizzare l'istruzione TND per il debug o la ricerca guasti eseguendo la logica solo fino ad un certo punto. Via via che viene eseguito il debug di una nuova sezione, spostare l'istruzione TND nella restante logica.

Se l'istruzione TND è abilitata, il controllore interrompe la scansione della routine corrente.

Ladder



Testo strutturato

```
TND();
```

Master Control Reset (MCR)

L'istruzione MCR, usata a coppie, crea una zona di programma in grado di disabilitare tutti i rami all'interno dell'istruzione MCR.

Operandi:

Ladder

nessuno

Descrizione: Se la zona MCR è abilitata, i rami della zona MCR vengono scanditi normalmente per verificare le condizioni di vero o falso. Se è disabilitata, il controllore scandisce comunque i rami della zona MCR, ma il tempo di scansione è ridotto poiché tutte le uscite non ritentive della zona sono disabilitate. La condizione del ramo di ingresso è falsa per tutte le istruzioni all'interno della zona MCR disabilitata.

Quando si programma una zona MCR, notare quanto segue:

- È necessario terminare la zona con un'istruzione MCR incondizionata.
- Non è possibile annidare una zona MCR all'interno di un'altra.
- Non saltare in una zona MCR. Se la zona è falsa, il saltare in questa la rende attiva a partire dal punto in cui si è saltati fino alla fine della zona.
- Se una zona MCR continua fino alla fine della routine, non è necessario programmare un'istruzione MCR per terminare la zona.

l'istruzione MCR non sostituisce il relè di controllo principale cablato per l'arresto di emergenza. Per garantire l'arresto di emergenza degli I/O, bisogna in ogni caso installare un relè di controllo principale cablato.

ATTENZIONE

Non sovrapporre o annidare zone MCR. Ogni zona MCR deve essere separata e completa. In caso contrario potrebbe verificarsi un funzionamento anomalo della macchina con possibili danni alle apparecchiature o alle persone.

Inserire le operazioni critiche al di fuori della zona MCR. Se si avvia un'istruzione, ad esempio un timer, in una zona MCR, l'esecuzione dell'istruzione si interrompe quando la zona è disabilitata ed il timer viene azzerato.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso. Le istruzioni della zona vengono scandite, ma la condizione del ramo di ingresso è falsa e le uscite non ritentive sono disabilitate.
se la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su vero. Le istruzioni della zona vengono scandite normalmente.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

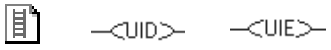
Esempio: Quando la prima istruzione MCR è abilitata (*input_1*, *input_2* e *input_3* impostati), il controllore esegue i rami nella zona MCR (tra le due istruzioni MCR) e, a seconda delle condizioni di ingresso, imposta o azzera le uscite.

Quando la prima istruzione MCR è disabilitata (*input_1*, *input_2* e *input_3* non sono tutti impostati), il controllore esegue i rami nella zona MCR (tra le due istruzioni MCR) e la condizione del ramo di ingresso diventa falsa per tutti i rami della zona MCR, a prescindere dalle condizioni di ingresso.

User Interrupt Disable (UID) User Interrupt Enable (UIE)

Le istruzioni UID e UIE funzionano congiuntamente per evitare che alcuni rami critici vengano interrotti da altri task.

Operandi:



Ladder

nessuno

```
UID();
UIE();
```

Testo strutturato

nessuno

È necessario inserire le parentesi ( ) dopo lo mnemonico dell'istruzione, anche se non ci sono operandi.

Descrizione: Quando la condizione del ramo di ingresso è vera:

- l'istruzione UID impedisce ai task con maggiore priorità di interrompere il task corrente ma *non* disabilita l'esecuzione di un routine di errore o del Programma di Gestione Errori del Controllore.
- l'istruzione UIE abilita altri task ad interrompere il task corrente.

Per evitare che una serie di rami venga interrotta:

1. Limitare il numero di rami che *non* si vuole vengano interrotti. La disabilitazione degli interrupt per un periodo di tempo prolungato può provocare un'interruzione delle comunicazioni.
2. Sopra al primo ramo che *non* si desidera venga interrotto, inserire un ramo ed un'istruzione UID.
3. Dopo l'ultimo ramo della serie che *non* si desidera che venga interrotto, inserire un ramo ed un'istruzione UIE.
4. Se richiesto, è possibile annidare coppie di istruzioni UID/UIE.

Indicatori di stato aritmetico: non influenzati

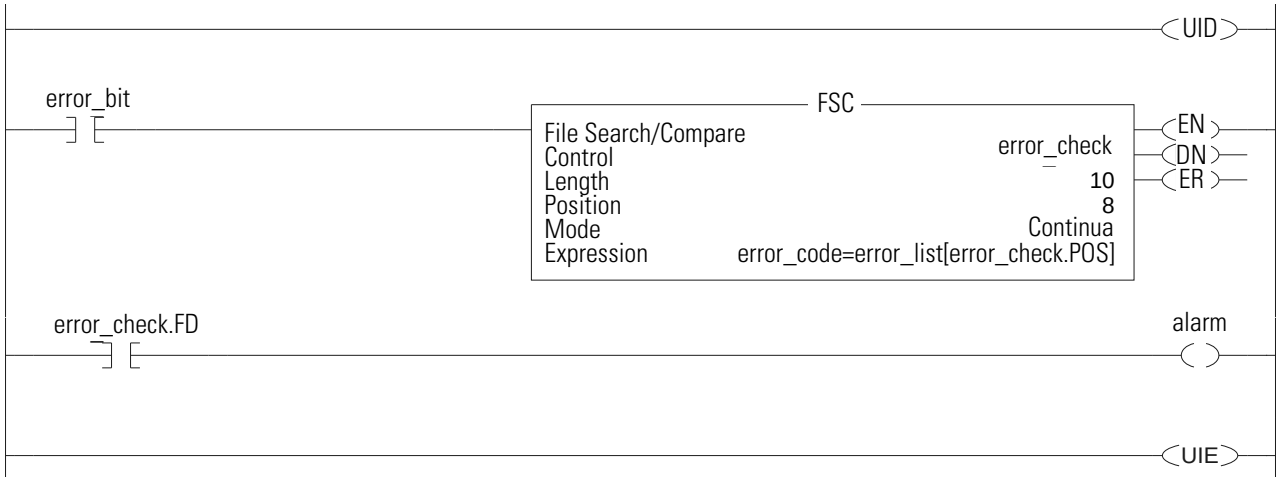
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione UID previene l'interrupt da parte di un task ad alta priorità. L'istruzione UIE abilita l'interrupt da parte di un task ad alta priorità.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Se si verifica un errore (*error_bit* è attivo), l'istruzione FSC verifica se il codice di errore è presente in un elenco di errori critici. Se l'istruzione FSC trova che l'errore è critico (error_check.FD è attivo), viene attivato un allarme. Le istruzioni UID e UIE evitano che altri task interrompano il controllo errori e l'allarme.

Ladder



Testo strutturato

```
UID();

    <istruzioni>

UIE();
```

Always False Instruction (AFI)

L'istruzione AFI imposta la sua condizione del ramo di uscita su falso.

Operandi:

Ladder

nessuna

Descrizione: L'istruzione AFI imposta la sua condizione del ramo di uscita su falso.

Indicatori di stato aritmetico: non influenzati

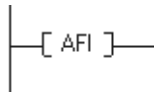
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su falso.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Utilizzare l'istruzione AFI per disabilitare temporaneamente un ramo mentre si esegue il debug di un programma.

Se è abilitata, l'istruzione AFI disabilita tutte le istruzioni di questo ramo.



No Operation (NOP)

L'istruzione NOP funziona come un marcatore di posizione

Operandi:

Ladder

nessuno

Descrizione: È possibile inserire l'istruzione NOP in una posizione qualsiasi del ramo. Se è abilitata, l'istruzione NOP non esegue alcuna operazione. Se è disabilitata, l'istruzione NOP non esegue alcuna operazione.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

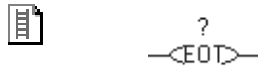
Esempio Questa istruzione è utile per localizzare diramazioni incondizionate inserendo un'istruzione NOP sulla diramazione.

L'istruzione NOP aggira l'istruzione XIC per abilitare l'uscita.

End of Transition (EOT)

L'istruzione EOT restituisce uno stato booleano in una transizione SFC.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
bit di dati	BOOL	tag	stato della transizione (0=in esecuzione, 1=completata)

```
EOT(data_bit);
```

Testo strutturato

Gli operandi sono gli stessi dell'istruzione EOT in logica ladder.

Descrizione:

L'istruzione EOT restituisce uno stato booleano, per cui è possibile che più routine SFC condividano la stessa routine contenente l'istruzione EOT. Se la routine chiamante non è una transizione, l'istruzione EOT agisce come un'istruzione TND (vedere pagina 444).

L'implementazione dell'istruzione EOT sui controllori Logix è diversa da quella dei controllori PLC-5. Nel caso di un controllore PLC-5 l'istruzione EOT non ha parametri, ma restituisce la condizione del ramo come stato. Invece nel caso del controllore Logix il parametro di ritorno restituisce lo stato della transizione poiché lo stato del ramo non è disponibile in tutti i linguaggi di programmazione Logix.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

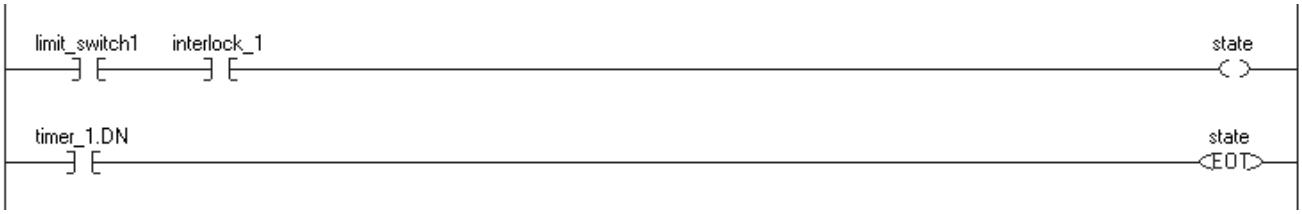
nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione restituisce il valore dei bit di dati alla routine chiamante.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Quando sono impostati sia *limit_switch1* che *interlock_1* impostare lo stato. Al termine di *timer_1*, EOT restituisce il valore dello *state* alla routine chiamante.

Ladder



Testo strutturato

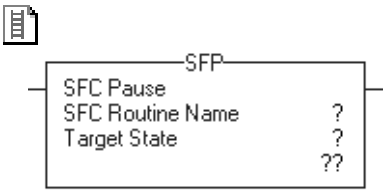
```
state := limit_switch1 AND interlock_1;


IF timer_1.DN THEN

     EOT(state);

END_IF;
```

SFC Pause (SFP)

L'istruzione consente di mettere in pausa una routine SFC.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
SFCRoutine Name	ROUTINE	nome	routine SFC da mettere in pausa
TargetState	DINT	immediato	selezionare:
		tag	eseguire (o inserire 0)
			pausa (o inserire 1)

Testo strutturato

```
SFP(SFCRoutineName,
    TargetState);
```

Gli operandi sono gli stessi dell'istruzione SFP in logica ladder.

Descrizione:

L'istruzione SFP consente di mettere in pausa una routine SFC in esecuzione. Quando la routine SFC è in pausa, si usa nuovamente l'istruzione SFP per modificarne lo stato e riprendere l'esecuzione.

Inoltre, l'istruzione SFP serve per riprendere l'esecuzione della SFC dopo aver utilizzato un'istruzione SFR (vedere pagina 456) per ripristinare una routine SFC.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
la routine non è di tipo SFC	4	85

Esecuzione:

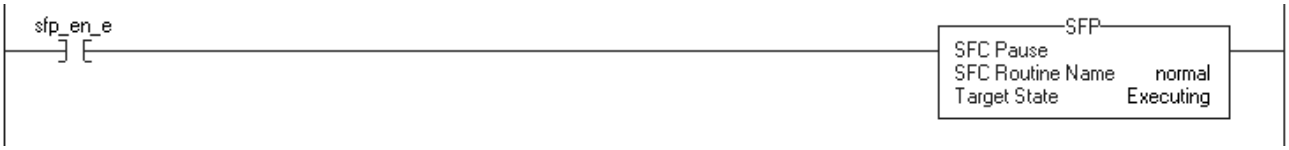
Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione mette in pausa o riprende l'esecuzione della routine SFC specificata.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Se *sfc_en_p* è impostato, mettere in pausa la routine SFC denominata *normal*. Riprendere l'esecuzione della SFC quando viene impostato *sfc_en_e*.

Ladder

Mettere in pausa la routine SFC.

Riprendere l'esecuzione della routine SFC.



Testo strutturato

Mettere in pausa la routine SFC:

```
IF (sfp_en_p) THEN
     SFP(normal,paused);
     sfp_en_p := 0;
END_IF;
```

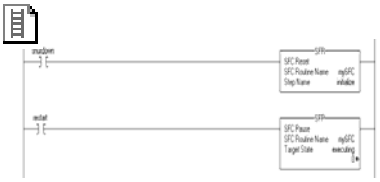
Riprendere l'esecuzione della routine SFC:

```
IF (sfp_en_e) THEN
     SFP(normal,executing);
     sfp_en_e := 0;
END_IF;
```

SFC Reset (SFR)

L'istruzione SFR ripristina l'esecuzione di una routine SFC in corrispondenza di un passo specificato.

Operandi:



Operandi ladder

Operando	Tipo	Formato	Descrizione
SFCRoutine Name	ROUTINE	nome	routine SFC da ripristinare
Step Name	SFC_STEP	tag	passo target da cui riprendere l'esecuzione

Testo strutturato

```
SFR(SFCRoutineName,StepName);
```

Gli operandi sono gli stessi dell'istruzione SFR in logica ladder.

Descrizione:

Quando l'istruzione SFR è abilitata:

- nella routine SFC specificata, tutte le azioni memorizzate smettono di essere eseguire (ripristino)
- l'istruzione SFC inizia a essere eseguita al passo specificato.

Se il passo target è 0, il diagramma verrà ripristinato al passo iniziale

L'implementazione dell'istruzione SFR sui controllori Logix è diversa da quella dei controllori PLC-5. Nel caso dei controllori PLC-5, la SFR veniva eseguita quando la condizione del ramo era vera. Dopo l'azzeramento la SFC rimaneva in pausa fino a quando il ramo contenente la SFR non diventava falso. In questo modo era possibile ritardare l'esecuzione in seguito a un ripristino. Questa funzione di pausa/riprendi dell'istruzione PLC-5 SFR è stata scissa dalla condizione del ramo e trasferita all'istruzione SFP.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
la routine non è di tipo SFC	4	85
il passo target specificato non esiste nella routine SFC	4	89

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione reimposta la routine SFC specificata.	L'istruzione reimposta la routine SFC specificata.
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Se si verifica una condizione specifica (è impostato *shutdown*), riavviare la SFC al passaggio *initialize*.

Ladder

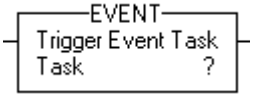
Testo strutturato

```
IF shutdown THEN

     SFR(mySFC,initialize);

END_IF;
```

Trigger Event Task (EVENT)

L'istruzione EVENT aziona l'esecuzione di un task evento.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Task	TASK	nome	task evento da eseguire L'istruzione consente di scegliere altri tipi di task, ma non li esegue.

```
EVENT(task_name);
```

Testo strutturato

Gli operandi sono gli stessi dell'istruzione EVENT in logica ladder.

Descrizione: Usare l'istruzione EVENT per eseguire via programma un task evento:

- Ogni volta che l'istruzione viene eseguita, attiva il task evento specificato.
- Accertarsi di dare al task evento tempo a sufficienza per completare la propria esecuzione prima di attivarlo nuovamente. In caso contrario, si verifica una sovrapposizione.
- Se si esegue un'istruzione EVENT mentre il task evento è già in esecuzione, il controllore aumenta il contatore di sovrapposizione ma non attiva il task evento.

Determinare via programma se un'istruzione EVENT ha azionato un task

Per determinare se un'istruzione EVENT ha azionato un task evento, usare un'istruzione Get System Value (GSV) per monitorare l'attributo Status del task.

Attributo Status dell'oggetto TASK

Attributo	Tipo di dati	Istruzione	Descrizione	
Status	DINT	GSV SSV	Fornisce informazioni sullo stato del task. Una volta che il controllore imposta un bit, occorre azzerarlo manualmente per determinare se si è verificato un altro errore di quel tipo.	
			Per determinare se:	**Esamina questo bit:**
			L'istruzione EVENT ha attivato il task (solo task evento).	*0*
			Un timeout ha attivato il task (solo task evento).	1
			Si è verificata una sovrapposizione per questo task.	2

Il controllore non azzera i bit dell'attributo Status una volta che sono stati impostati.

- Per usare un bit per nuove informazioni sullo stato, occorre azzerare manualmente il bit.
- Usare un'istruzione Imposta valore di sistema (SSV) per impostare l'attributo su un valore diverso.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione EVENT attiva l'esecuzione del task evento specificato	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

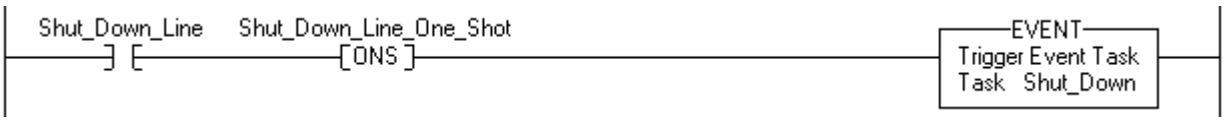
Esempio 1: Un controllore usa più programmi ma una procedura di spegnimento comune. Ogni programma usa un tag del programma chiamato *Shut_Down_Line* che si attiva se il programma rileva una condizione che richiede lo spegnimento. La logica in ciascun programma viene eseguita nel modo seguente:

Se *Shut_Down_Line* = on (condizioni che richiedono uno spegnimento), allora

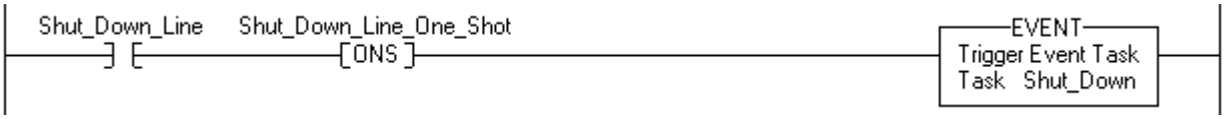
Eseguire il task *Shut_Down* una volta

Ladder

Programma A



Programma B



Testo strutturato

Programma A

```
IF Shut_Down_Line AND NOT Shut_Down_Line_One_Shot THEN
  EVENT (Shut_Down);
END_IF;
Shut_Down_Line_One_Shot := Shut_Down_Line;
```

Programma B

```
IF Shut_Down_Line AND NOT Shut_Down_Line_One_Shot THEN
  EVENT (Shut_Down);
END_IF;
Shut_Down_Line_One_Shot := Shut_Down_Line;
```

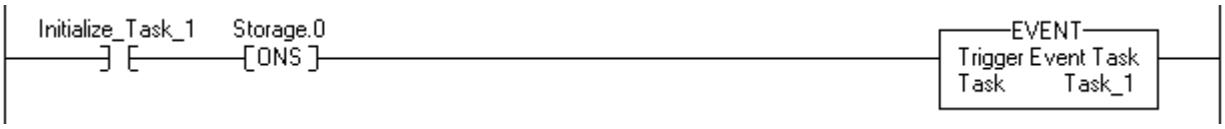
Esempio 2: L'esempio seguente usa un'istruzione EVENT per inizializzare un task evento (un altro tipo di evento attiva normalmente il task evento).

Task continuo

Se *Initialize_Task_1* = 1 allora

L'istruzione ONS limita l'esecuzione dell'istruzione EVENT a una scansione.

L'istruzione EVENT attiva l'esecuzione di *Task_1* (task evento).

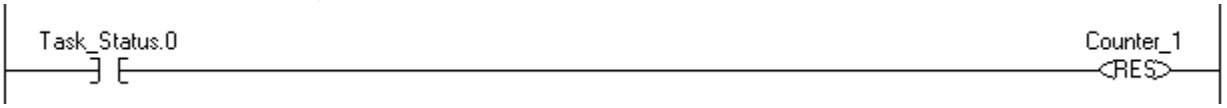


Task_1 (task evento)

L'istruzione GSV imposta *Task_Status* (DINT tag) = attributo Status per il task evento. Nell'attributo Instance Name, THIS indica l'oggetto TASK per il task in cui si trova l'istruzione (ossia *Task_1*).

Se *Task_Status*.0 = 1 allora un'istruzione EVENT ha attivato il task evento (ossia quando il task continuo è in esecuzione, la sua istruzione EVENT inizializza il task evento).

L'istruzione RES ripristina un contatore utilizzato dal task evento.

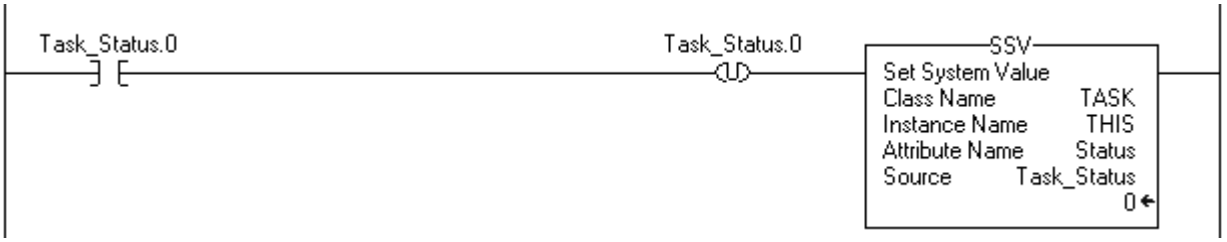


Il controllore non azzera i bit dell'attributo Status una volta che sono stati impostati. Per usare un bit per nuove informazioni sullo stato, occorre azzerare manualmente il bit.

Se *Task_Status*.0 = 1 azzerare il bit.

L'istruzione OTU imposta *Task_Status*.0 = 0.

L'istruzione SSV imposta l'attributo Status del task THIS (*Task_1*) = *Task_Status*. Questo comprende il bit azzerato.



Note:

Istruzioni For/Break

(FOR, FOR...DO, BRK, EXIT, RET)

Introduzione

Utilizzare l'istruzione FOR per richiamare ripetutamente una subroutine. Utilizzare l'istruzione BRK per interrompere l'esecuzione di una subroutine.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
Eseguire ripetutamente una routine.	FOR	ladder	464
	FOR...DO(1)	testo strutturato	
Terminare l'esecuzione ripetuta di una routine.	BRK	ladder	467
	EXIT(1)	testo strutturato	
Ritornare all'istruzione FOR.	RET	ladder	468

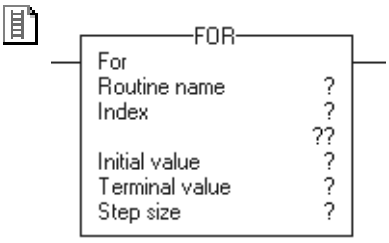
(1) Solo testo strutturato.

For (FOR)

L'istruzione FOR esegue una subroutine ripetutamente.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Routine name	ROUTINE	nome routine	routine da eseguire
Index	DINT	tag	calcola quante volte è stata eseguita la routine
Initial value	SINT	immediato	valore iniziale dell'indice
	INT	tag	
	DINT		
Terminal value	SINT	immediato	valore a cui interrompere l'esecuzione della routine
	INT	tag	
	DINT		
Step size	SINT	immediato	valore da sommare all'indice ogni volta che l'istruzione FOR esegue la routine
	INT	tag	
	DINT		

Testo strutturato

```
FOR conteggio:=
valore_iniziale TO
valore_finale BY incremento
DO
```

Usare il costrutto FOR...DO. Vedere l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi dei costrutti in testo strutturato.

Descrizione:

IMPORTANTE

Non utilizzare un'istruzione FOR per richiamare (eseguire) la routine principale.

- Si può inserire un'istruzione FOR nella routine principale o in qualunque altra routine.
- Se si utilizza un'istruzione FOR per richiamare la routine principale e quindi si inserisce un'istruzione RET nella routine principale, si verifica un errore grave (tipo 4, codice 31).

Se è abilitata, l'istruzione FOR esegue ripetutamente la routine fino a quando il valore Index non supera il Terminal value. Questa istruzione non passa parametri alla routine.

Ogni volta che l'istruzione FOR esegue la routine essa aggiunge il valore Step size all'Index.

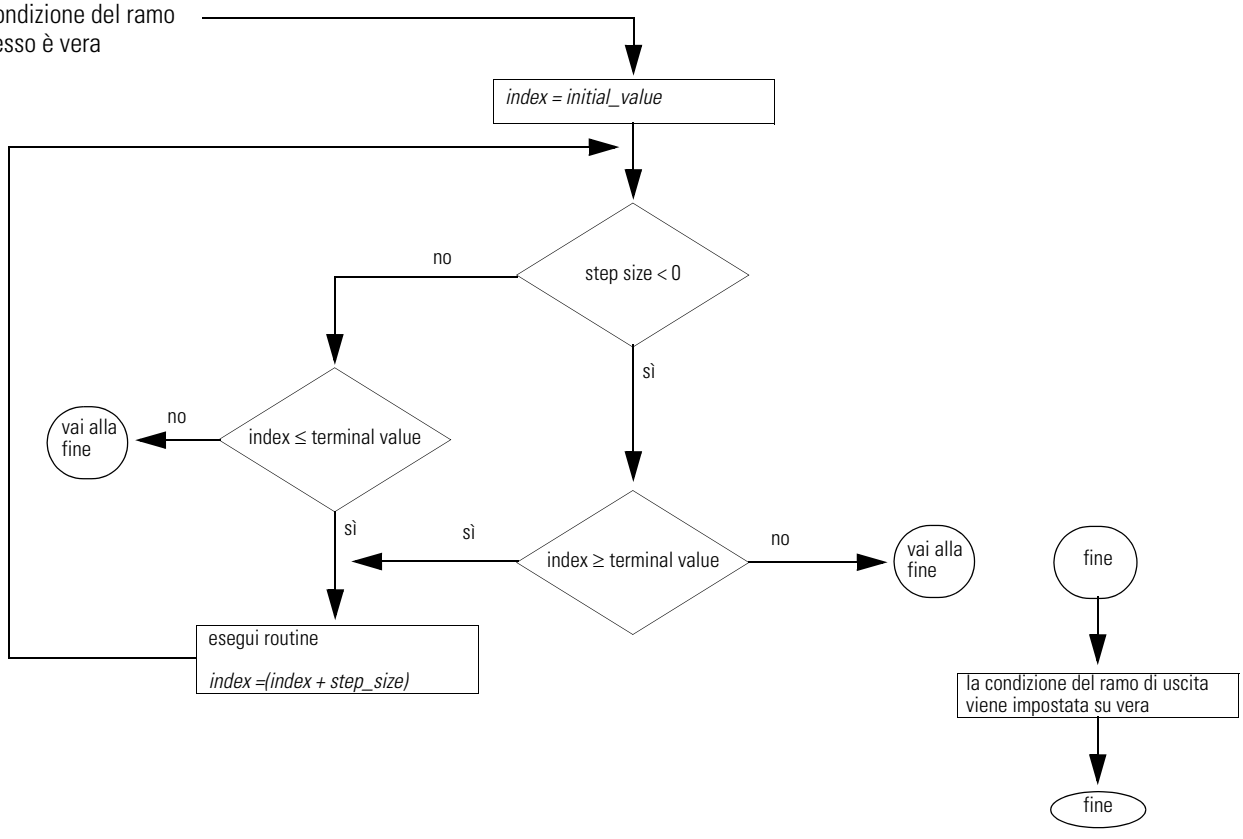
Fare attenzione a non creare troppe iterazioni in una singola scansione. Un numero eccessivo di ripetizioni può provocare il timeout del watchdog del controllore e quindi un errore grave.

Indicatori di stato aritmetico: non influenzati

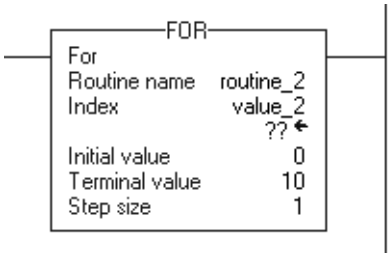
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
la routine principale contiene un'istruzione RET	4	31

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso. Il controllore esegue la subroutine una volta. Se vi sono delle istruzioni FOR ricorsive che richiamano la stessa subroutine, questa viene prescansionata solo la prima volta. Se vi sono più istruzioni FOR (non ricorsive) che richiamano la stessa subroutine, questa viene prescansionata ogni volta.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

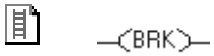
Esempio: Se è abilitata, l'istruzione FOR esegue ripetutamente la *routine_2* e, ogni volta, fa incrementare *value_2* di 1. Quando *value_2* è >10 oppure se è abilitata un'istruzione BRK, l'istruzione FOR non esegue più la *routine_2*.



Break (BRK)

L'istruzione BRK interrompe l'esecuzione di una routine richiamata da un'istruzione FOR.

Operandi:



Ladder

nessuna

```
EXIT;
```

Testo strutturato

L'istruzione EXIT viene usata in un ciclo. Vedere l'Appendice B per informazioni sulla sintassi dei costrutti in testo strutturato.

Descrizione: Se è abilitata, l'istruzione BRK termina la routine e riporta il controllore all'istruzione seguente alla FOR.

Se vi sono delle istruzioni FOR annidate, l'istruzione BRK riporta il controllo all'istruzione FOR più interna.

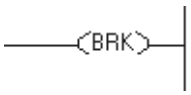
Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su vero. L'esecuzione ritorna all'istruzione successiva all'istruzione FOR chiamante.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione BRK interrompe l'esecuzione della routine corrente e ritorna all'istruzione successiva all'istruzione FOR chiamante.

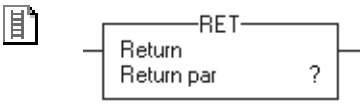


Return (RET)

L'istruzione RET riporta all'istruzione FOR chiamante.

Operandi:

Ladder



nessuna

Descrizione:

IMPORTANTE Non inserire un'istruzione RET nella routine principale. Se si inserisce un'istruzione RET nella routine principale, si verifica un errore grave (tipo 4, codice 31).

Se è abilitata, l'istruzione RET riporta all'istruzione FOR. L'istruzione FOR fa incrementare il valore Index del valore Step size ed esegue nuovamente la subroutine. Se il valore Index supera il Terminal value, l'istruzione FOR viene completata e l'esecuzione continua con l'istruzione seguente l'istruzione FOR.

L'istruzione FOR non utilizza parametri. L'istruzione FOR ignora qualsiasi parametro inserito in un'istruzione RET.

Per terminare l'esecuzione di una subroutine è possibile anche utilizzare un'istruzione TND.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

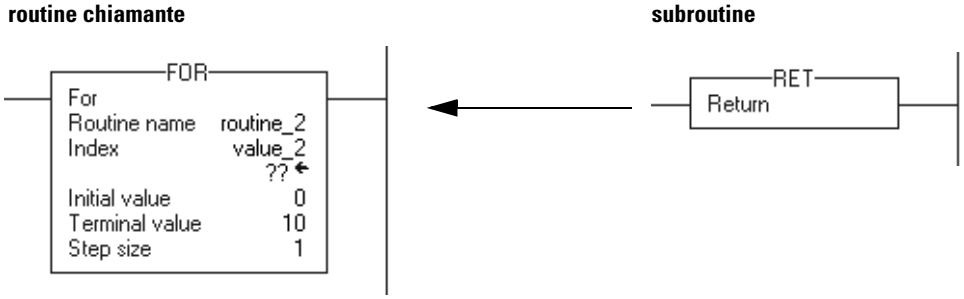
Si verifica un errore grave se	Tipo errore	Codice errore
la routine principale contiene un'istruzione RET	4	31

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Riporta i parametri specificati alla routine chiamante. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: L'istruzione FOR esegue ripetutamente la *routine_2* e, ogni volta, fa incrementare *value_2* di 1. Quando *value_2* è > 10 oppure se è abilitata un'istruzione BRK, l'istruzione FOR non esegue più la *routine_2*.

L'istruzione RET riporta all'istruzione FOR chiamante. L'istruzione FOR esegue nuovamente la subroutine e fa incrementare il valore Index del valore Step size, oppure, se il valore Index supera il Terminal value, l'istruzione viene completata e l'esecuzione continua con l'istruzione seguente all'istruzione FOR.



Note:

Istruzioni speciali
(FBC, DDT, DTR, PID)

Introduzione

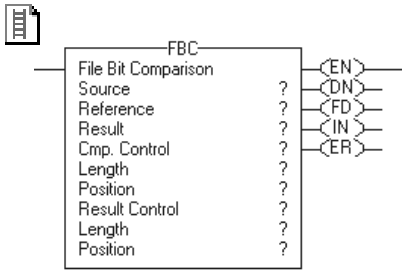
Le istruzioni speciali eseguono delle operazioni specifiche a seconda delle applicazioni.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
Confrontare i dati con un file di dati di riferimento noto ed affidabile e registrare le eventuali non corrispondenze.	FBC	ladder	472
Confrontare i dati con un file di dati di riferimento noto ed affidabile e registrare le eventuali non corrispondenze e aggiornare il riferimento in funzione della Source.	DDT	ladder	479
Far passare i dati Source attraverso una maschera e confrontare il risultato con i dati di riferimento. Quindi scrivere la Source nel file di riferimento per effettuare il confronto successivo.	DTR	ladder	486
Controllare un anello PID.	PID	ladder testo strutturato	489

File Bit Comparison (FBC)

L'istruzione FBC confronta i bit della matrice Source con i bit di una matrice Reference.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source	DINT	tag matrice	matrice da confrontare al riferimento **non** usare CONTROL.POS nell'indice
Reference	DINT	tag matrice	matrice da confrontare alla Source **non** usare CONTROL.POS nell'indice
Result	DINT	tag matrice	matrice dove memorizzare il risultato **non** usare CONTROL.POS negli indici
Cmp control	CONTROL	struttura	struttura di controllo per il confronto
Length	DINT	immediato	numero di bit da confrontare
Position	DINT	immediato	posizione corrente nella source il valore iniziale generalmente è 0
Result control	CONTROL	struttura	struttura di controllo dei risultati
Length	DINT	immediato	numero di locazioni di memorizzazione nel risultato
Position	DINT	immediato	posizione corrente nel risultato il valore iniziale generalmente è 0

ATTENZIONE

Usare tag diversi per la struttura di controllo del confronto e la struttura di controllo del risultato.
Se si utilizza lo stesso tag si potrebbe verificare un funzionamento anomalo della macchina con possibili danni alle apparecchiature o alle persone.

Struttura COMPARE

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione FBC è abilitata.
.DN	BOOL	Il bit di fine viene impostato quando l'istruzione FBC confronta l'ultimo bit nelle matrici Source e Riferimento.
.FD	BOOL	Il bit di trovato viene impostato ogni volta che l'istruzione FBC rileva una mancata corrispondenza (modalità "uno alla volta") o dopo avere registrato tutte le mancate corrispondenze (modalità "tutti").
.IN	BOOL	Il bit di inibizione indica la modalità di ricerca FBC. 0 = Modalità Tutti 1 = Modalità Uno alla volta
.ER	BOOL	Il bit di errore viene impostato se il.POS di confronto < 0, il.LEN di confronto < 0, il .POS risultato < 0 o il.LEN risultato < 0. L'esecuzione dell'istruzione viene interrotta fino a quando il programma non cancella il bit .ER.
.LEN	DINT	Il valore length indica il numero di bit da confrontare.
.POS	DINT	Il valore posizione indica il bit corrente.

Struttura RESULT

Mnemonico	Tipo di dati	Descrizione
.DN	BOOL	Il bit di fine viene impostato quando la matrice Result è piena.
.LEN	DINT	Il valore Length indica il numero di posizioni di memorizzazione nella matrice Result.
.POS	DINT	Il valore di posizione specifica la posizione corrente nella matrice Result.

Descrizione: Se è abilitata, l'istruzione FBC confronta i bit della matrice Source con i bit della matrice Reference e registra nella matrice Result il numero di bit di ogni mancata corrispondenza.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione FBC opera su una memoria contigua. In alcuni casi, le istruzioni cercano o scrivono oltre la matrice in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

La differenza tra l'istruzione DDT e FBC consiste nel fatto che ogni volta che un'istruzione DDT trova una mancata corrispondenza, l'istruzione cambia il bit di riferimento facendolo corrispondere con il bit Source. L'istruzione FBC non cambia il bit di riferimento.

Scelta della modalità di ricerca

Se si vuole rilevare	Selezionare questa modalità
Una mancata corrispondenza per volta	Impostare il bit .IN nella struttura di controllo del confronto. Ogni volta che la condizione del ramo di ingresso passa da falsa a vera, l'istruzione FBC cerca la successiva mancata corrispondenza tra le matrici Source e Reference. Quando trova una mancata corrispondenza, l'istruzione imposta il bit .FD, registra la posizione della mancata corrispondenza ed interrompe l'esecuzione.
Tutte le mancate corrispondenze	Azzerare il bit .IN nella struttura di controllo del confronto. Ogni volta che la condizione del ramo di ingresso passa da falsa a vera, l'istruzione FSC cerca tutte le mancate corrispondenze tra le matrici Source e Reference.

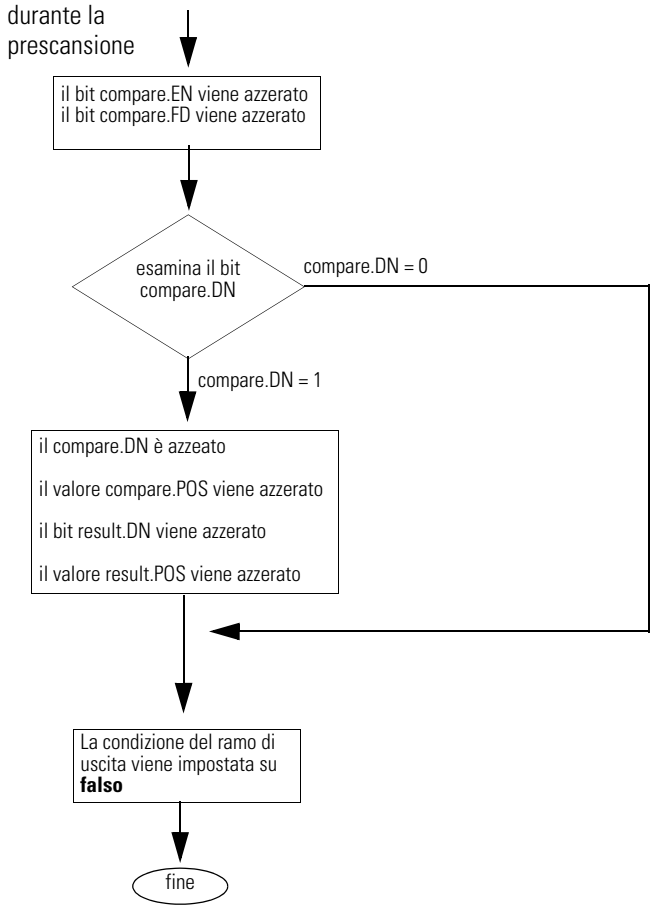
Indicatori di stato aritmetico: non influenzati

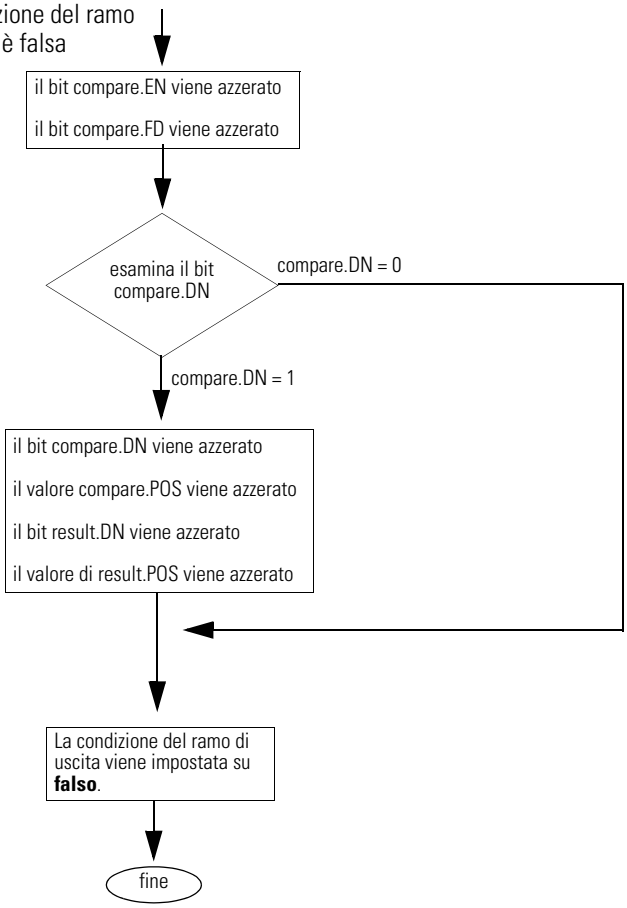
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
Result.POS > dimensione della matrice Result	4	20

Esecuzione:

Condizione	Azione ladder

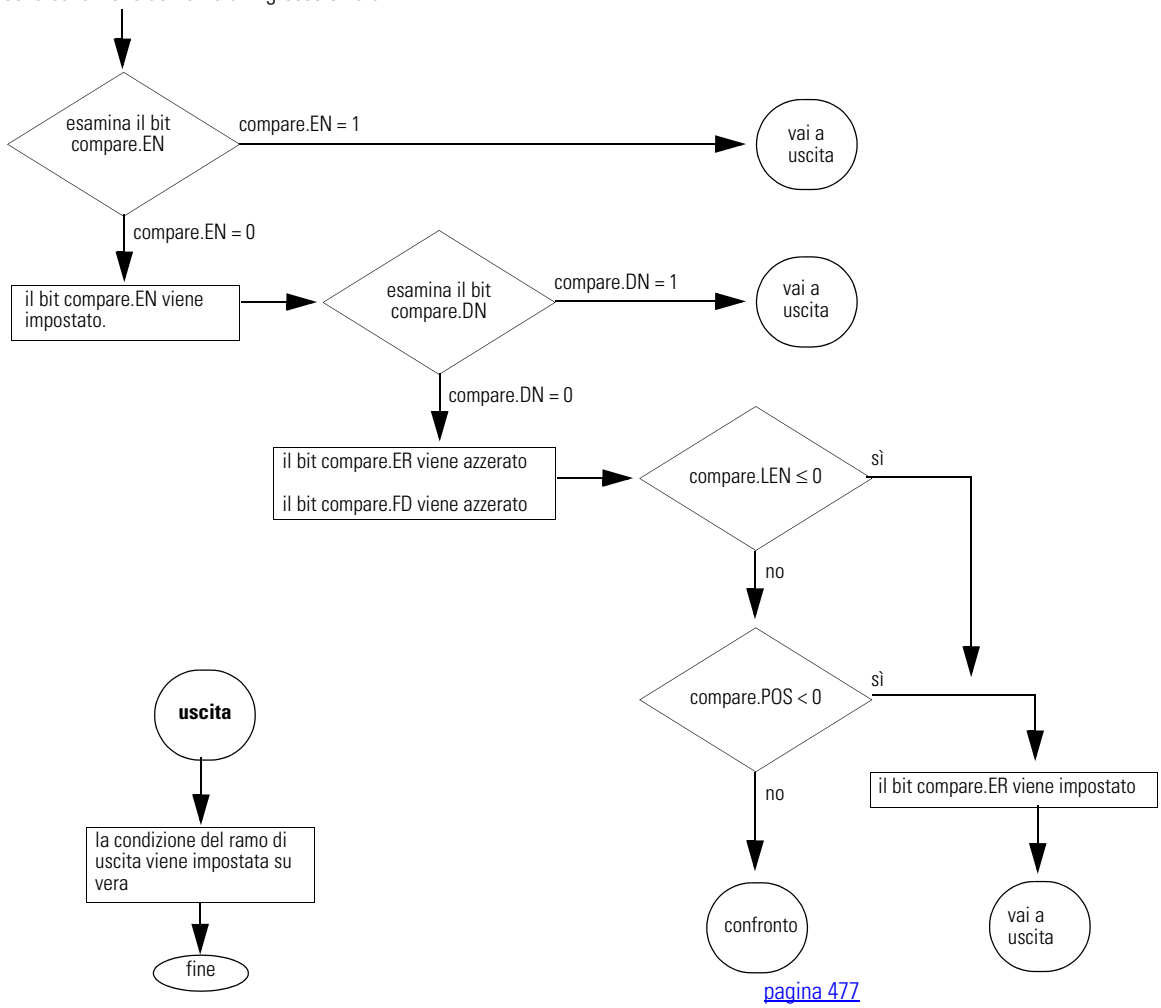

durante la prescansione
il bit compare.EN viene azzerato
il bit compare.FD viene azzerato
esamina il bit compare.DN
compare.DN = 0
compare.DN = 1
il compare.DN è azzeato
il valore compare.POS viene azzerato
il bit result.DN viene azzerato
il valore result.POS viene azzerato
La condizione del ramo di uscita viene impostata su **falso**
fine

Condizione	Azione ladder
se la condizione del ramo di ingresso è falsa	

il bit compare.EN viene azzerato
il bit compare.FD viene azzerato
esamina il bit compare.DN
compare.DN = 0
compare.DN = 1
il bit compare.DN viene azzerato
il valore compare.POS viene azzerato
il bit result.DN viene azzerato
il valore di result.POS viene azzerato
La condizione del ramo di uscita viene impostata su **falso**.
fine

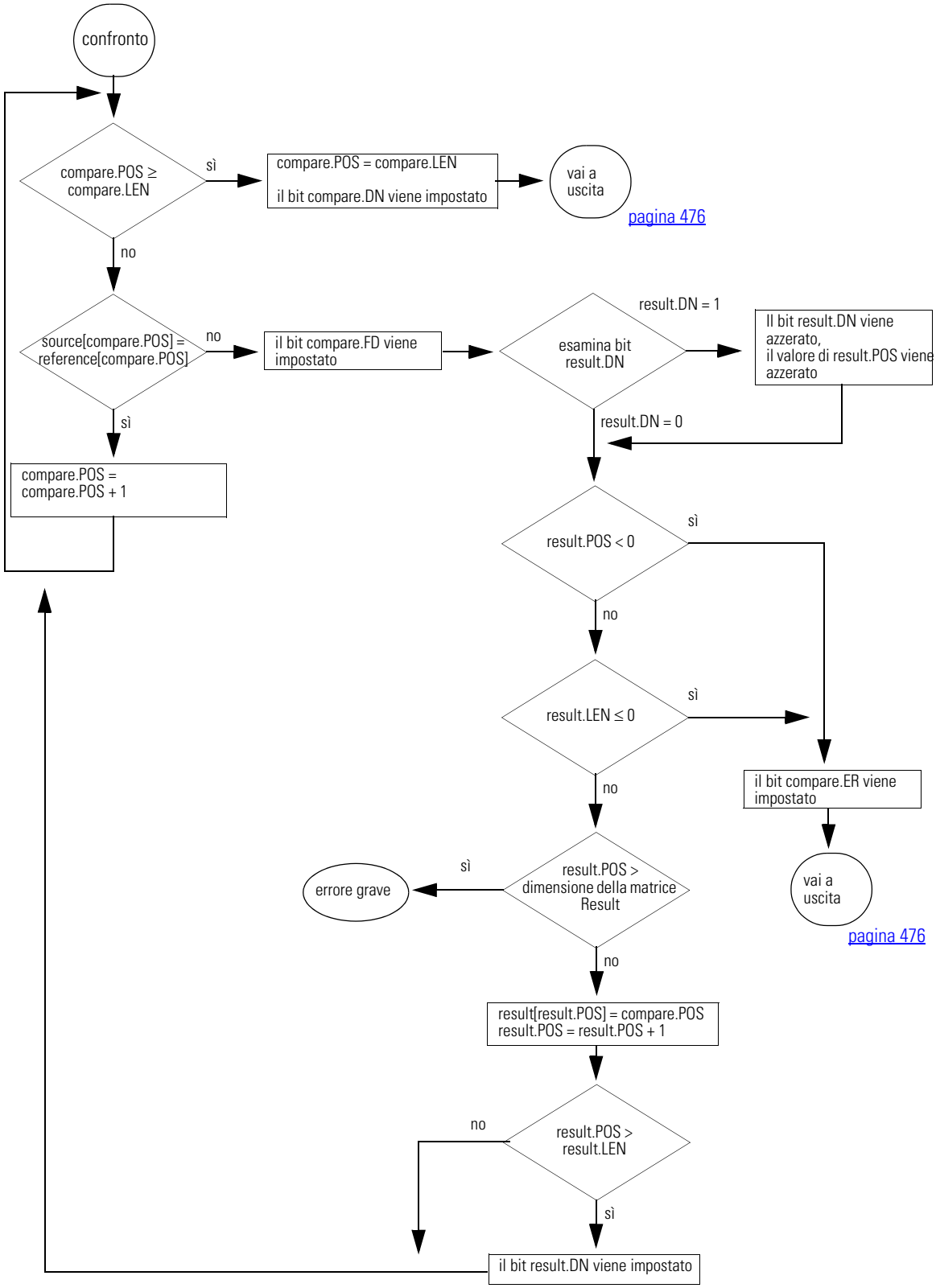
Condizione	Azione ladder

se la condizione del ramo di ingresso è vera



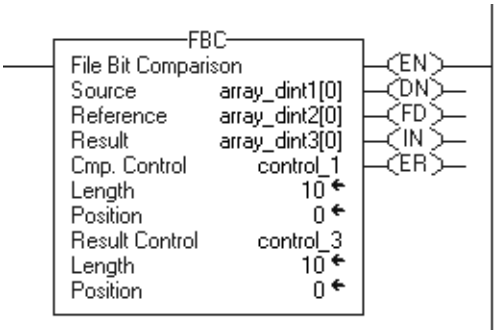
pagina 477

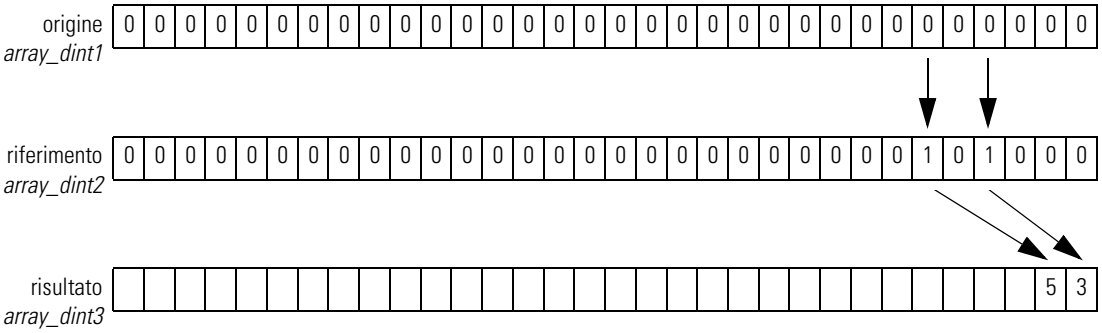
Condizione	Azione ladder



post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione FBC confronta *array_dint1* di origine con la *array_dint2* di riferimento e memorizza le posizioni delle mancate corrispondenze nella *array_dint3* di risultato.

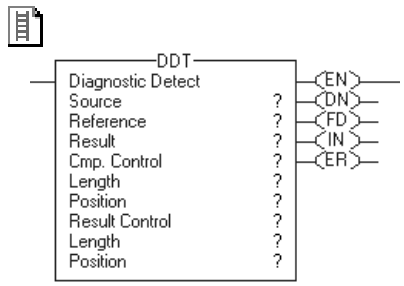




Diagnostic Detect (DDT)

L'istruzione DDT confronta i bit della matrice Source con i bit della matrice Reference per determinare eventuali cambiamenti di stato.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source	DINT	tag matrice	matrice da confrontare al riferimento **non** usare CONTROL.POS nell'indice
Reference	DINT	tag matrice	matrice da confrontare alla Source **non** usare CONTROL.POS nell'indice
Result	DINT	tag matrice	matrice dove memorizzare i risultati **non** usare CONTROL.POS nell'indice
Cmp control	CONTROL	struttura	struttura di controllo per il confronto
Length	DINT	immediato	numero di bit da confrontare
Position	DINT	immediato	posizione corrente nella source il valore iniziale generalmente è 0
Result control	CONTROL	struttura	struttura di controllo dei risultati
Length	DINT	immediato	numero di locazioni di memorizzazione nel risultato
Position	DINT	immediato	posizione corrente nel risultato il valore iniziale generalmente è 0

ATTENZIONE

Usare tag diversi per la struttura di controllo del confronto e la struttura di controllo del risultato. !
Se si utilizza lo stesso tag si potrebbe verificare un funzionamento anomalo della macchina con possibili danni alle apparecchiature o alle persone.

Struttura COMPARE

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione DDL è abilitata.
.DN	BOOL	Il bit di fine viene impostato quando l'istruzione DDT confronta l'ultimo bit nelle matrici di origine e riferimento.
.FD	BOOL	Il bit di trovato viene impostato ogni volta che l'istruzione DDT rileva una mancata corrispondenza (modalità "uno alla volta") o dopo avere registrato tutte le mancate corrispondenze (modalità "tutti").
.IN	BOOL	Il bit di inibizione indica la modalità di ricerca DDT. 0 = Modalità Tutti 1 = Modalità Uno alla volta
.ER	BOOL	Il bit di errore viene impostato se il.POS di confronto < 0, il.LEN di confronto < 0, il .POS risultato < 0 o il.LEN risultato < 0. L'esecuzione dell'istruzione viene interrotta fino a quando il programma non cancella il bit .ER.
.LEN	DINT	Il valore Length indica il numero di bit da confrontare.
.POS	DINT	Il valore posizione indica il bit corrente.

Struttura RESULT

Mnemonico	Tipo di dati	Descrizione
.DN	BOOL	Il bit di fine viene impostato quando la matrice Result è piena.
.LEN	DINT	Il valore Length indica il numero di posizioni di memorizzazione nella matrice Result.
.POS	DINT	Il valore di posizione specifica la posizione corrente nella matrice Result.

Descrizione: Se è abilitata, l'istruzione DDT confronta i bit della matrice Source con i bit della matrice Reference, registra nella matrice Result il numero di bit di ogni mancata corrispondenza e cambia il valore del bit di Reference in modo che corrisponda con il valore del relativo bit di Source.

IMPORTANTE È necessario testare e confermare che le istruzioni non modifichino i dati che non si desidera cambiare.

L'istruzione DDT opera su una memoria contigua. In alcuni casi, le istruzioni cercano o scrivono oltre la matrice in altri membri del tag. Questo avviene se la lunghezza è eccessiva e il tag è un tipo di dati definito dall'utente.

La differenza tra l'istruzione DDT e FBC consiste nel fatto che ogni volta che un'istruzione DDT trova una mancata corrispondenza, l'istruzione cambia il bit di riferimento in modo che corrisponda con il bit della Source. L'istruzione FBC non cambia il bit di riferimento.

Scelta della modalità di ricerca

Se si vuole rilevare	Selezionare questa modalità
Una mancata corrispondenza per volta	Impostare il bit .IN nella struttura di controllo del confronto. Ogni volta che la condizione del ramo di ingresso passa da falsa a vera, l'istruzione DDT cerca la successiva mancata corrispondenza tra le matrici Source e Reference. Quando trova una mancata corrispondenza, l'istruzione imposta il bit .FD, registra la posizione della mancata corrispondenza ed interrompe l'esecuzione.
Tutte le mancate corrispondenze	Azzerare il bit .IN nella struttura di controllo del confronto. Ogni volta che la condizione del ramo di ingresso passa da falsa a vera, l'istruzione DDT cerca tutte le mancate corrispondenze tra le matrici Source e Reference.

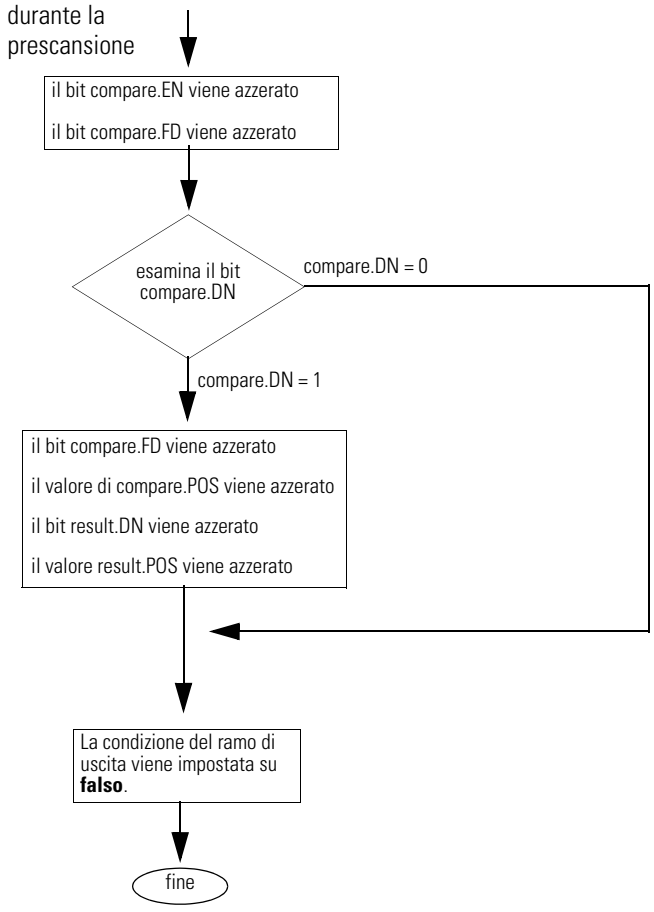
Indicatori di stato aritmetico: non influenzati

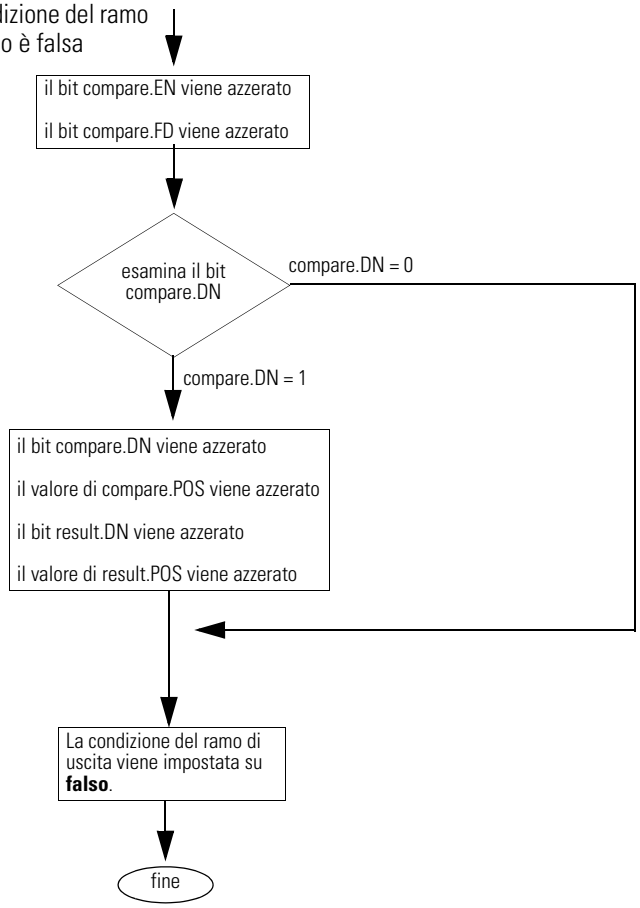
Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
Result.POS > dimensione della matrice Result	4	20

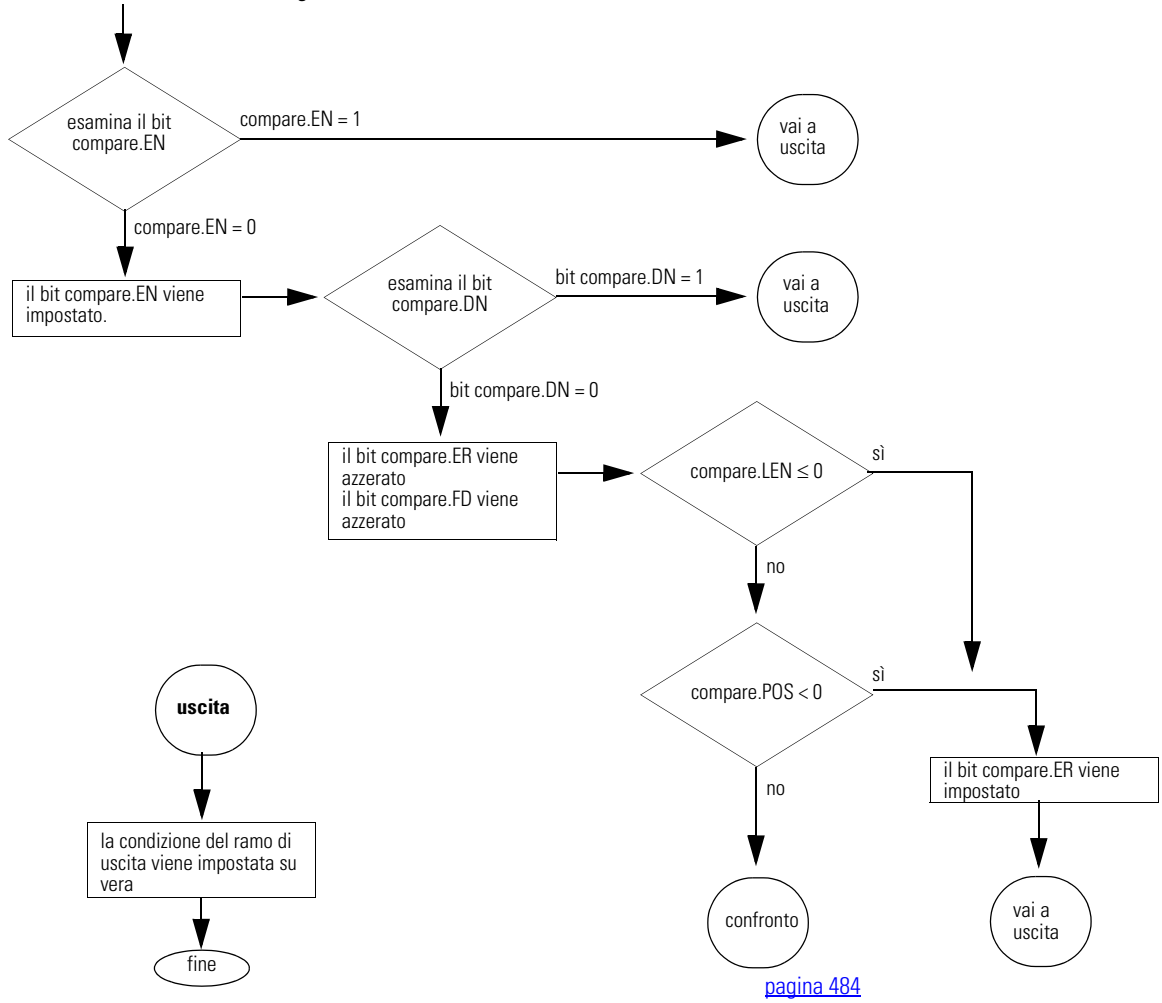
Esecuzione:

Condizione	Azione ladder


durante la prescansione
il bit compare.EN viene azzerato
il bit compare.FD viene azzerato
esamina il bit compare.DN
compare.DN = 0
compare.DN = 1
il bit compare.FD viene azzerato
il valore di compare.POS viene azzerato
il bit result.DN viene azzerato
il valore result.POS viene azzerato
La condizione del ramo di uscita viene impostata su **falso**.
fine

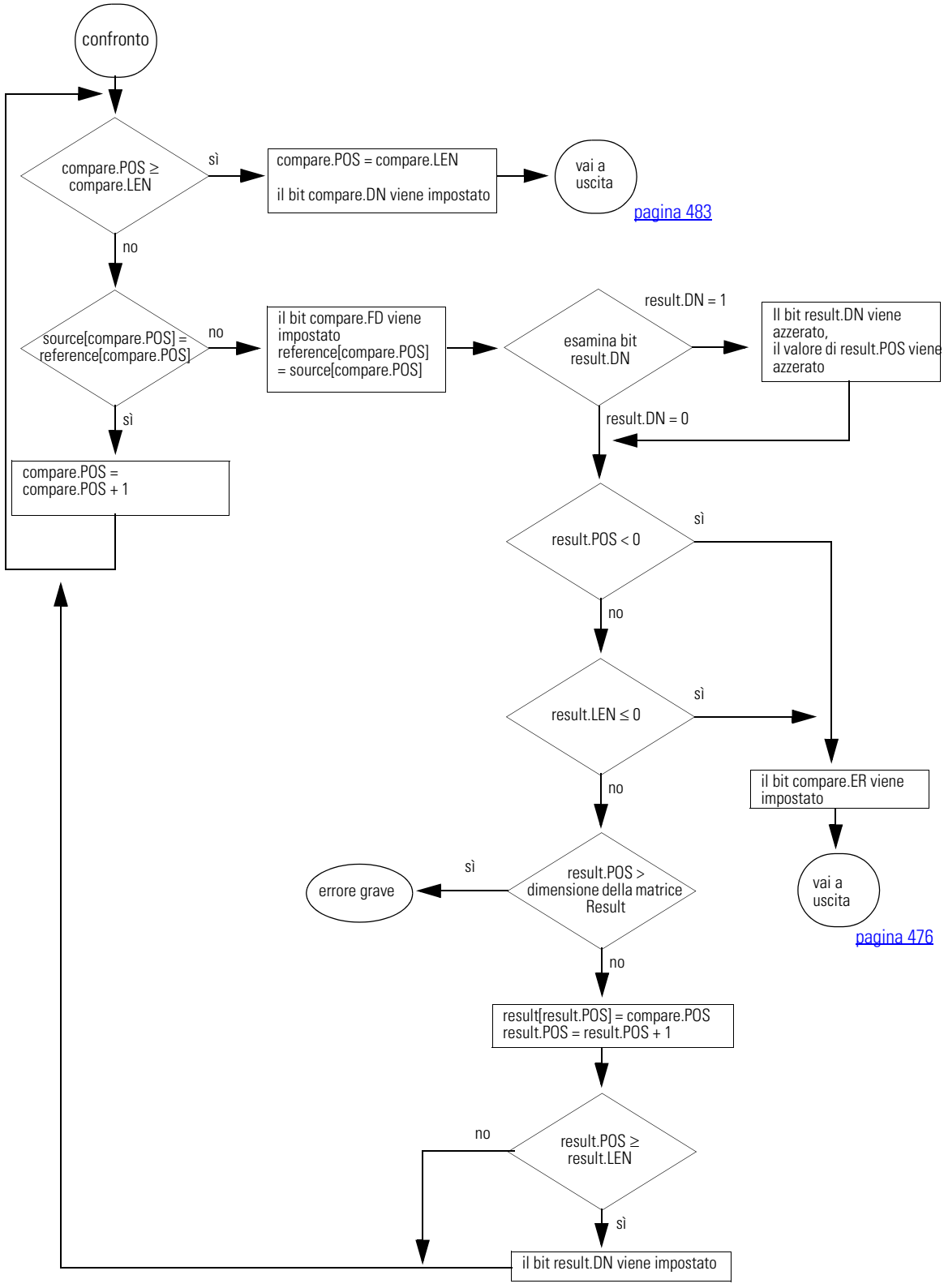
Condizione	Azione ladder
se la condizione del ramo di ingresso è falsa	

il bit compare.EN viene azzerato
il bit compare.FD viene azzerato
esamina il bit compare.DN
compare.DN = 0
compare.DN = 1
il bit compare.DN viene azzerato
il valore di compare.POS viene azzerato
il bit result.DN viene azzerato
il valore di result.POS viene azzerato
La condizione del ramo di uscita viene impostata su **falso**.
fine

Condizione	Azione ladder
se la condizione del ramo di ingresso è vera	

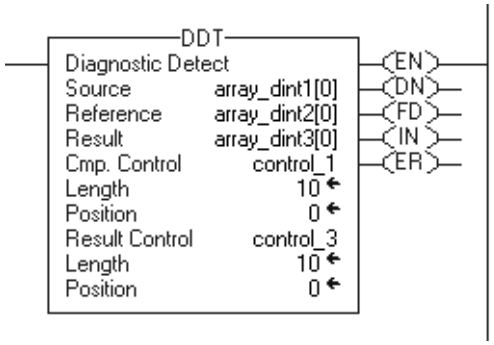
pagina 484

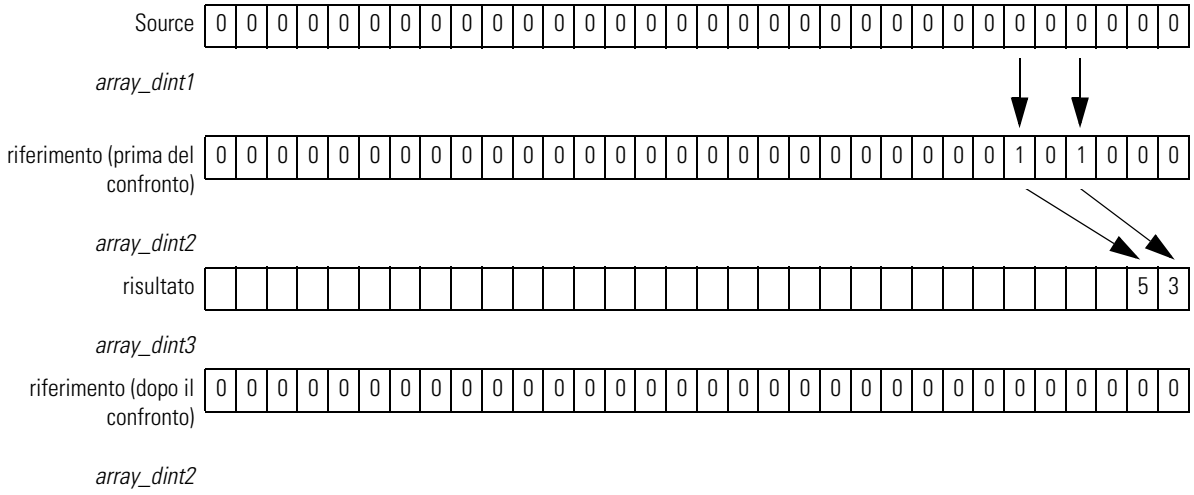
Condizione	Azione ladder



post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Se è abilitata, l'istruzione DDT confronta *array_dint1* della Source con la *array_dint2* di riferimento e memorizza le posizioni delle mancate corrispondenze nella *array_dint3* di risultato. Il controllore cambia anche i bit non corrispondenti di *array_dint2* del riferimento in modo che corrispondano a quelli di *array_dint1* della Source.



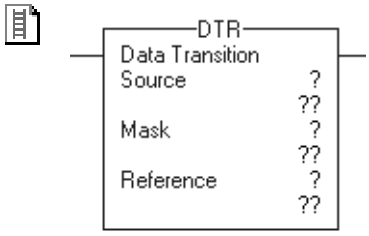

Source
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
array_dint1
riferimento (prima del confronto)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
array_dint2
risultato
5 3
array_dint3
riferimento (dopo il confronto)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
array_dint2

Data Transitional (DTR)

L'istruzione DTR passa il valore Source attraverso una maschera e confronta il risultato con il valore Reference.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	DINT	immediato tag	matrice da confrontare al riferimento
Mask	DINT	immediato tag	quali bit bloccare o fare passare
Reference	DINT	tag	matrice da confrontare a Source

Descrizione:

L'istruzione DTR passa il valore Source attraverso una maschera e confronta il risultato con il valore Reference. L'istruzione DTR, inoltre, scrive il valore Source mascherato nel valore Reference per il successivo confronto. Il valore Source rimane invariato.

Un "1" nella maschera significa che il bit dati viene fatto passare. Uno "0" nella maschera significa che il bit dati è bloccato.

Se il valore Source con maschera è diverso dal valore Reference, la condizione del ramo di uscita diventa vera per una scansione. Se il valore Source con maschera e il valore Reference sono gli stessi, la condizione del ramo di uscita è falsa.

ATTENZIONE

È pericoloso programmare online con questa istruzione. Se il valore Reference è diverso dal valore Source, la condizione del ramo di uscita diventa vera. Fare attenzione nell'inserire questa istruzione quando il processore si trova in modalità Run o Remote Run.

Immissione di un valore di maschera immediato

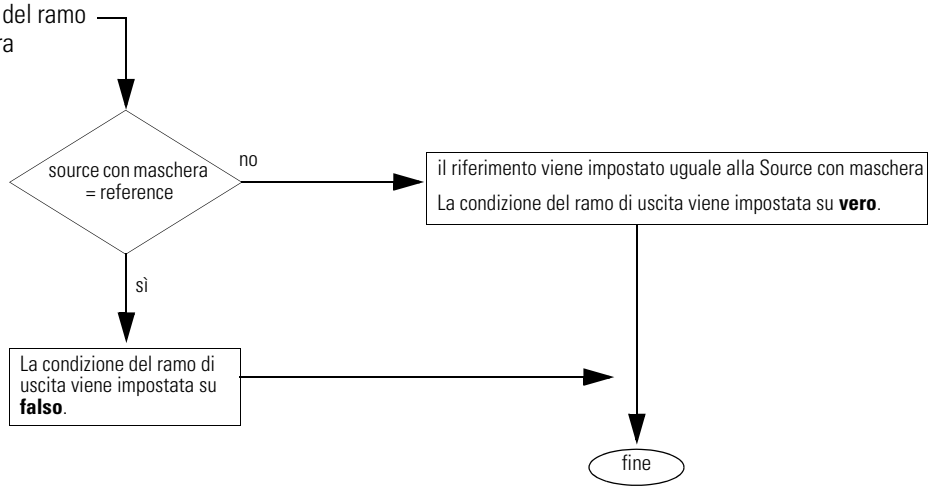
Quando si immette una maschera, il software di programmazione imposta automaticamente valori decimali. Se si desidera una maschera utilizzando un altro formato, fare precedere il prefisso corretto al valore.

Prefisso	Descrizione
16#	Esadecimale per esempio; 16#0F0F
8#	Ottale per esempio; 8#16
2#	Binario per esempio; 2#00110011

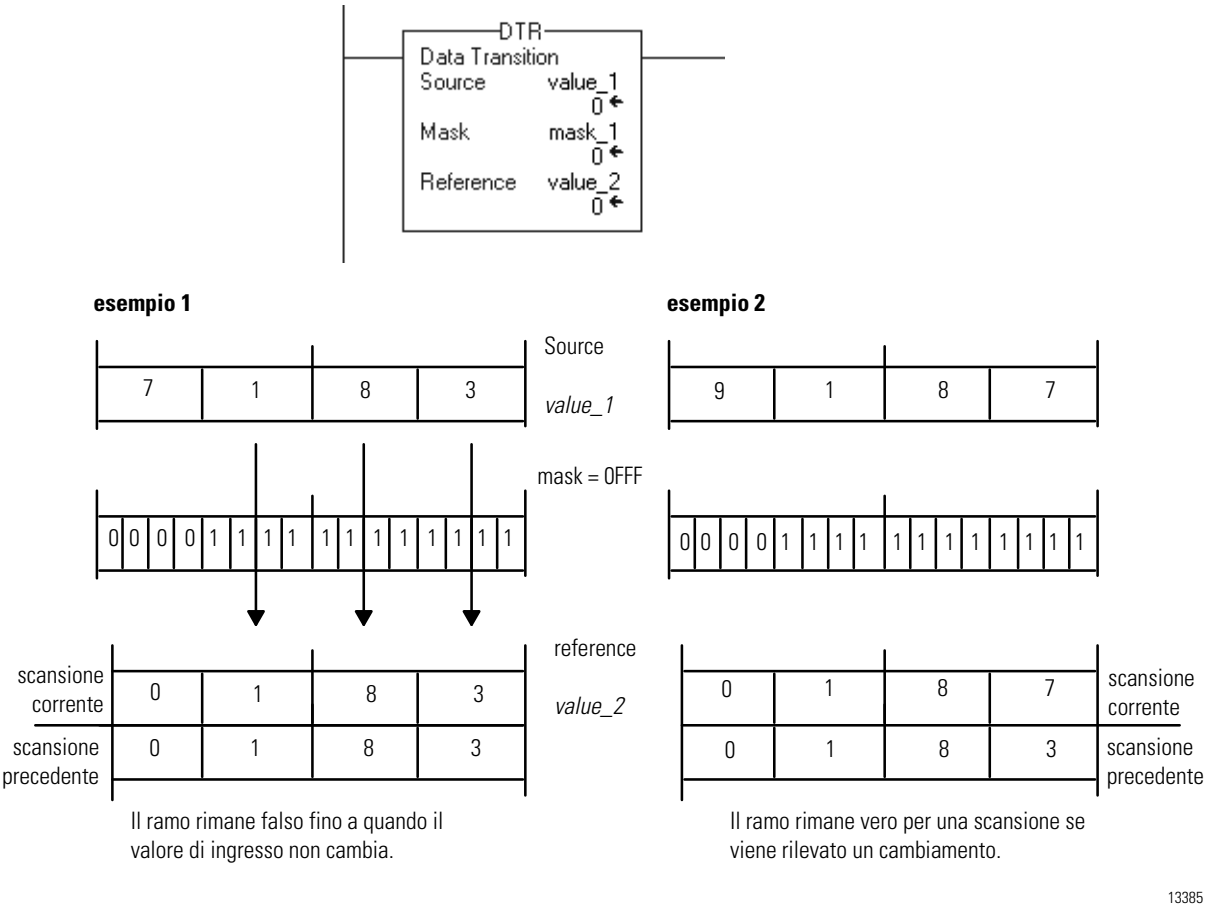
Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	Reference = Source AND Mask. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	Reference = Source AND Mask. La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Quando è abilitata, l'istruzione DTR maschera *value_1*. Se c'è differenza tra i due valori, la condizione del ramo di uscita viene impostata a vera.



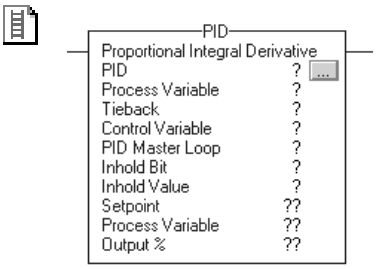
Uno 0 nella maschera lascia il bit invariato.

Proportional Integral Derivative (PID)

L'istruzione PID controlla una variabile di processo come ad esempio flusso, pressione, temperatura o livello.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
PID	PID	struttura	struttura PID
Process variable	SINT INT DINT REAL	tag	valore che si vuole controllare
Tieback	SINT INT DINT REAL	immediato tag	*(opzionale)* uscita di una stazione manuale/automatica hardware che bypassa l'uscita del controllore Immettere 0 se non si desidera utilizzare questo parametro.
Control variable	SINT INT DINT REAL	tag	valore che viene inviato al dispositivo di controllo finale (valvola, damper, ecc.) Se si utilizza la banda morta, la variabile di controllo deve essere REAL altrimenti essa verrà forzata a 0 se l'errore è all'interno della banda morta.
PID master loop	PID	struttura	(*opzionale*) tag PID per il PID master Se si esegue un controllo a cascata e questo PID è un anello slave, inserire il nome del PID master. Immettere 0 se non si desidera utilizzare questo parametro.
Inhold bit	BOOL	tag	(*opzionale*) stato corrente del bit inhold di un canale di uscita analogico 1756 per il supporto del riavviamento graduale Immettere 0 se non si desidera utilizzare questo parametro.
Inhold value	SINT INT DINT REAL	tag	(*opzionale*) valore di ritorno dati da un canale di uscita analogico 1756 per il supporto del riavviamento graduale Immettere 0 se non si desidera utilizzare questo parametro.

Operando	Tipo	Formato	Descrizione
Setpoint			visualizzazione valore corrente del setpoint
Process variable			visualizzazione valore corrente della variabile di processo in scala
Output %			visualizzazione valore in percentuale dell'uscita corrente

```
PID(PID,ProcessVariable,
Tieback,ControlVariable,
PIDMasterLoop,InholdBit,
InHoldValue);
```

Testo strutturato

Gli operandi sono gli stessi dell'istruzione PID in logica ladder. Tuttavia, i valori Setpoint, Process Variable, e Output % vengono indicati accedendo ai membri .SP, .PV. e .OUT della struttura PID invece di inserire il valore nella lista di operandi.

Struttura PID

Mnemonico	Tipo di dati	Descrizione	
.CTL	DINT	Il membro .CTL permette di accedere ai membri di stato (bit) in una parola a 32 bit. L'istruzione PID imposta i bit 07–15.	
		Questo bit:	**È questo membro:**
		31	.EN
		30	.CT
		29	.CL
		28	.PVT
		27	.DOE
		26	.SWM
		25	.CA
		24	.MO
		23	.PE
		22	.NDF
		21	.NOBC
		20	.NOZC
		Questo bit:	**È questo membro impostato dall'istruzione PID:**
		15	.INI
		14	.SPOR
		13	.OLL
		12	.OLH
		11	.EWD
		10	.DVNA
		09	.DVPA
		08	.PVLA
		07	.PVHA
.SP	REAL	setpoint	
.KP	REAL	indipendente	guadagno proporzionale (adimensionale)
		dipendente	guadagno del controllore (adimensionale)

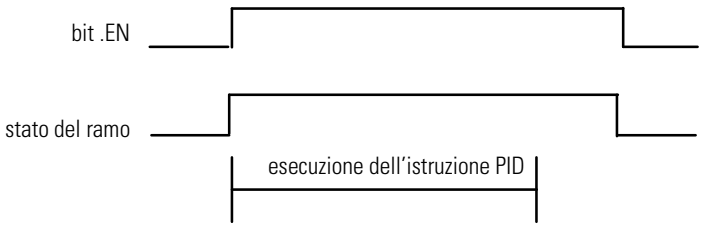
Mnemonico	Tipo di dati	Descrizione	
.KI	REAL	indipendente	guadagno integrale (1/sec)
		dipendente	tempo integrale (minuti per ripetizione)
.KD	REAL	indipendente	guadagno derivativo (secondi)
		dipendente	tempo derivativo (minuti)
.BIAS	REAL	% della compensazione anticipata o del bias	
.MAXS	REAL	fondo scala superiore in unità ingegneristiche	
.MINS	REAL	fondo scala inferiore in unità ingegneristiche	
.DB	REAL	banda morta in unità ingegneristiche	
.SO	REAL	% di uscita impostata	
.MAXO	REAL	limite uscita massimo (% dell'uscita)	
.MINO	REAL	limite uscita minimo (% dell'uscita)	
.UPD	REAL	tempo di aggiornamento dell'anello (secondi)	
.PV	REAL	valore PV scalato	
.ERR	REAL	valore errore scalato	
.OUT	REAL	output %	
.PVH	REAL	limite superiore allarme della variabile di processo	
.PVL	REAL	limite inferiore allarme della variabile di processo	
.DVP	REAL	limite allarme della deviazione positiva	
.DVN	REAL	limite allarme della deviazione negativa	
.PVDB	REAL	banda morta dell'allarme della variabile di processo	
.DVDB	REAL	banda morta dell'allarme della deviazione	
.MAXI	REAL	valore PV massimo (ingresso non in scala)	
.MINI	REAL	valore PV minimo (ingresso non in scala)	
.TIE	REAL	.TIE REAL valore di tieback per il controllo manuale	
.MAXCV	REAL	valore CV massimo (equivalente al 100 %)	
.MINCV	REAL	valore CV minimo (equivalente allo 0 %)	
.MINTIE	REAL	valore tieback minimo (equivalente al 100 %)	
.MAXTIE	REAL	valore tieback massimo (equivalente allo 0 %)	

Mnemonico	Tipo di dati	Descrizione	
.DATA	REAL[17]	Il membro .DATA memorizza:	
		Elemento:	**Descrizione**
		.DATA[0]	accumulazione integrale
		.DATA[1]	valore temporaneo dell' attenuazione derivativa
		.DATA[2]	valore .PV precedente
		.DATA[3]	valore .ERR precedente
		.DATA[4]	valore .SP precedente valido
		.DATA[5]	costante di conversione in scala della percentuale
		.DATA[6]	costante di conversione in scala della .PV
		.DATA[7]	costante di conversione in scala della derivativa
		.DATA[8]	valore .KP precedente
		.DATA[9]	valore .KI precedente
		.DATA[10]	valore .KD precedente
		.DATA[11]	guadagno dipendente .KP
		.DATA[12]	guadagno dipendente .KI
		.DATA[13]	guadagno dipendente .KD
		.DATA[14]	valore .CV precedente
		.DATA[15]	costante di riconversione dalla scala della .CV
		.DATA[16]	costante di riconversione dalla scala del tieback
.EN	BOOL	abilitato	
.CT	BOOL	tipo a cascata (0=slave; 1=master)	
.CL	BOOL	anello a cascata (0=no; 1=sì)	
.PVT	BOOL	tracciabilità variabile di processo (0=no; 1=sì)	
.DOE	BOOL	derivata di (0=PV; 1=errore)	
.SWM	BOOL	modalità manuale software (0=no-auto; 1=sì- manuale sw)	
.CA	BOOL	azione di controllo (0 indica E=SP-PV; 1 indica E=PV-SP)	
.MO	BOOL	modalità stazione (0=automatica; 1=manuale)	
.PE	BOOL	equazione PID (0=indipendente; 1=dipendente)	
.NDF	BOOL	nessuna attenuazione derivata (0=filtro di attenuazione derivata abilitato; 1=filtro di attenuazione derivata disabilitato)	
.NOBC	BOOL	nessun calcolo bias (0=calcolo bias abilitato; 1=calcolo bias disabilitato)	
.NOZC	BOOL	nessuna banda morta all' attraversamento dello zero (0=banda morta con attraversamento dello zero; 1=banda morta senza attraversamento dello zero)	
.INI	BOOL	PID inizializzato (0=no; 1=sì)	
.SPOR	BOOL	setpoint fuori gamma (0=no; 1=sì)	
.OLL	BOOL	CV al di sotto del limite minimo di uscita (0=no; 1=sì)	
.OLH	BOOL	CV al di sopra del limite massimo di uscita (0=no; 1=sì)	
.EWD	BOOL	errore all'interno della banda morta (0=no; 1=sì)	
.DVNA	BOOL	deviazione con allarme basso (0=no; 1=sì)	

Mnemonico	Tipo di dati	Descrizione
.DVPA	BOOL	deviazione con allarme alto (0=no; 1=sì)
.PVLA	BOOL	PV con allarme basso (0=no; 1=sì)
.PVHA	BOOL	PV con allarme alto (0=no; 1=sì)

Descrizione: In genere, l'istruzione PID riceve la variabile di processo (PV) da un modulo di ingresso analogico e modula un'uscita di variabile di controllo (CV) su un modulo di uscita analogico al fine di mantenere la variabile di processo sul valore di riferimento desiderato (setpoint).

Il bit .EN indica lo stato dell'esecuzione. Il bit .EN viene impostato quando la condizione del ramo di ingresso passa da falsa a vera. Quando la condizione del ramo di ingresso diventa falsa, il bit .EN viene azzerato. L'istruzione PID non utilizza un bit .DN e viene eseguita ogni scansione fino a quando la condizione del ramo di ingresso rimane vera.



Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

IMPORTANTE Questi errori erano di tipo grave sul controllore PLC-5.

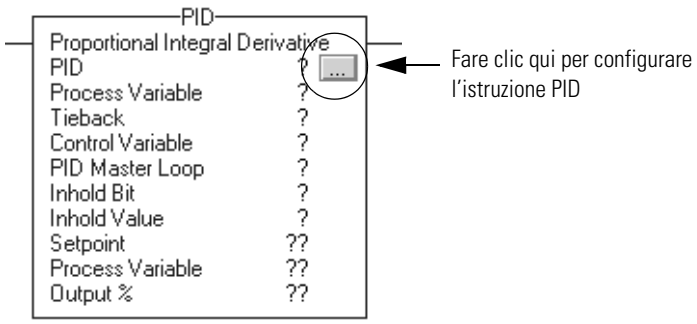
Si verifica un errore minore se	Tipo errore	Codice errore
.UPD ≤ 0	4	35
setpoint fuori gamma	4	36

Esecuzione:

Condizione	Azione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione esegue l'anello PID.	L'istruzione esegue l'anello PID.
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Configurazione di un'istruzione PID

Dopo avere immesso l'istruzione PID e specificato la struttura PID, utilizzare le schede di configurazione per specificare in che modo l'istruzione deve funzionare.



Messa a punto (Tuning)

Selezionare la scheda Tuning. Le modifiche hanno effetto dal momento in cui si fa clic su un altro campo, su OK, su Applica o quando si preme Invio.

In questo campo	Specificare
Setpoint (SP)	Inserire un valore di setpoint (.SP).
% di uscita impostata	Inserire una percentuale di uscita (.SO). In modalità Manuale software, questo valore viene utilizzato per l'uscita. In modalità Auto, questo valore visualizza la % dell'uscita
Output bias	Inserire una percentuale del bias di uscita (.BIAS).
Proportional gain (K_p)	Inserire il guadagno proporzionale (.KP). Per i guadagni indipendenti, è il guadagno proporzionale (adimensionale). Per i guadagni dipendenti, è il guadagno del controllore (adimensionale).
Integral gain (K_i)	Inserire il guadagno integrale (.KI). Per i guadagni indipendenti, è il guadagno integrale (1/sec). Per i guadagni dipendenti, è il tempo di azzeramento (minuti per ripetizione).
Derivative time (K_d)	Inserire il guadagno derivativo (.KD). Per i guadagni indipendenti, è il guadagno derivativo (secondi). Per i guadagni dipendenti, è il tempo derivativo (minuti).
Manual mode	Selezionare Manuale (.MO) o Manuale software (.SWM). Se si selezionano entrambe le modalità, la modalità Manuale sarà quella prevalente.

Configurazione

Selezionare la scheda Configuration. Fare clic su OK o Applica affinché i cambiamenti abbiano effetto.

In questo campo	Specificare
PID equation	Selezionare guadagni indipendenti o guadagni dipendenti (.PE). Scegliere Indipendente se si desidera che i tre guadagni (P, I e D) operino indipendentemente. Scegliere Dipendente se si desidera un guadagno generale del controllore che influenzi tutti e tre i termini (P, I e D).
Control action	Selezionare E=PV-SP oppure E=SP-PV per l'azione di controllo (.CA).
Derivative of	Selezionare PV o errore (.DOE). Utilizzare la derivata della PV per eliminare i picchi transitori derivanti dalle modifiche del setpoint. Utilizzare la derivata dell'errore per una rapida risposta alle modifiche del setpoint se l'algoritmo è in grado di tollerare sovraelongazioni.

In questo campo	Specificare
Loop update time	Immettere il tempo di aggiornamento (.UPD) dell'istruzione.
CV high limit	Inserire un limite superiore per la variabile di controllo (.MAXO).
CV low limit	Inserire un limite inferiore per la variabile di controllo (.MINO).
Deadband value	Inserire un valore per la banda morta (.DB).
No derivative smoothing	Abilitare o disabilitare questa voce (.NDF).
No bias calculation	Abilitare o disabilitare questa voce (.NOBC).
No zero crossing in deadband	Abilitare o disabilitare questa voce (.NOZC).
PV tracking	Abilitare o disabilitare questa voce (.PVT).
Cascade loop	Abilitare o disabilitare questa voce (.CL).
Cascade type	Se l'anello in cascata è abilitato, selezionare slave o master (.CT).

Allarmi

Selezionare la scheda Alarms. Fare clic su OK o Applica affinché i cambiamenti abbiano effetto.

In questo campo	Specificare
PV high	Inserire un valore di allarme alto di PV (.PVH).
PV low	Inserire un valore di allarme basso di PV (.PVL).
PV deadband	Inserire un valore di banda morta dell'allarme di PV (.PVBD).
positive deviation	Inserire un valore di deviazione positiva (.DVP).
negative deviation	Inserire un valore di deviazione negativa (.DVN).
deviation deadband	Inserire un valore di banda morta dell'allarme della deviazione (.DVDB).

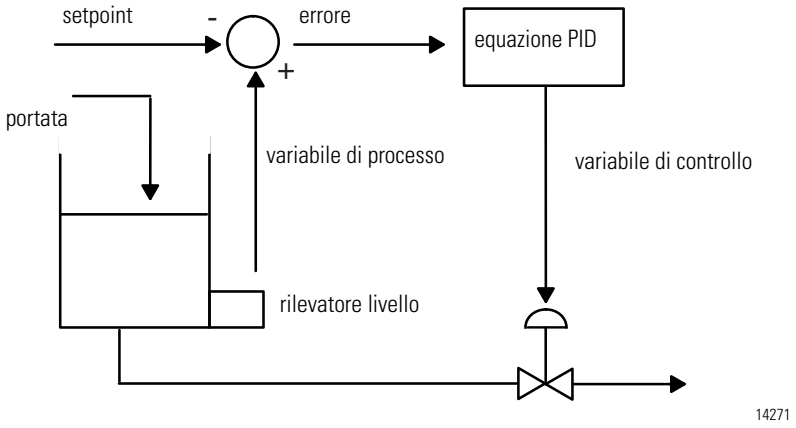
Conversione in scala

Selezionare la scheda Scaling. Fare clic su OK o Applica affinché i cambiamenti abbiano effetto.

In questo campo	Specificare
PV unscaled maximum	Inserire il valore massimo di PV (.MAXI) uguale al valore massimo non in scala ricevuto dal canale di ingresso analogico per il valore PV.
PV unscaled minimum	Inserire il valore minimo di PV (.MINI) uguale al valore minimo non in scala ricevuto dal canale di ingresso analogico per il valore PV.
PV engineering units maximum	Inserire il valore massimo in unità ingegneristiche corrispondente a .MAXI (.MAXS)
PV engineering units minimum	Inserire il valore minimo in unità ingegneristiche corrispondente a .MINI (.MINS)
CV maximum	Inserire il valore massimo di CV corrispondente al 100 % (.MAXCV).
CV minimum	Inserire il valore minimo di CV corrispondente allo 0 % (.MINCV).
Tieback maximum	Inserire il valore massimo di tieback (.MAXTIE) uguale al valore massimo non in scala ricevuto dal canale di ingresso analogico per il valore di tieback.
Tieback minimum	Inserire il valore minimo di tieback (.MINTIE) uguale al valore minimo non in scala ricevuto dal canale di ingresso analogico per il valore di tieback.
PID Initialized	Se si modificano le costanti di conversione in scala durante la modalità Run, disattivare questa opzione per inizializzare di nuovo i valori di riconversione interni (.INI).

Uso delle istruzioni PID

Il controllo ad anello chiuso PID mantiene una variabile di processo al valore di riferimento (setpoint) desiderato. La seguente figura mostra un esempio relativo alla portata/livello di fluidi:



In questo esempio, il livello del serbatoio viene confrontato con un valore di riferimento o setpoint. Se il livello è superiore al setpoint, l'equazione PID fa aumentare la variabile di controllo e provoca l'apertura della valvola di scarico del serbatoio facendo quindi diminuire il livello.

L'equazione PID utilizzata nell'istruzione PID è un'equazione posizionale con la possibilità di utilizzo di guadagni indipendenti o dipendenti. Quando si utilizzano i guadagni indipendenti, i guadagni proporzionali, integrali e derivativi influenzano solo i relativi termini specifici proporzionale, integrale o derivativo. Quando si utilizzano i guadagni dipendenti, il guadagno proporzionale è sostituito da un guadagno del controllore che influenza tutti e tre i termini. Per eseguire lo stesso tipo di controllo è possibile utilizzare indifferentemente una delle due forme di equazione. L'utente può scegliere l'equazione che preferisce.

Opzioni di guadagno	Derivata di	Equazione
Guadagni dipendenti (standard ISA)	errore (E)	$$CV = K_C\left[E + \frac{1}{T_i}\int_0^t Edt + T_d\frac{dE}{dt}\right] + BIAS$$
	Variabile di processo (PV)	E = SP – PV $$CV = K_C\left[E + \frac{1}{T_i}\int_0^t Edt - T_d\frac{dPV}{dt}\right] + BIAS$$ E = PV – SP $$CV = K_C\left[E + \frac{1}{T_i}\int_0^t Edt + T_d\frac{dPV}{dt}\right] + BIAS$$
Guadagni indipendenti	errore (E)	$$CV = K_PE + K_i\int_0^t Edt + K_d\frac{dE}{dt} + BIAS$$
	Variabile di processo (PV)	E = SP – PV $$CV = K_PE + K_i\int_0^t Edt - K_d\frac{dPV}{dt} + BIAS$$ E = PV – SP $$CV = K_PE + K_i\int_0^t Edt + K_d\frac{dPV}{dt} + BIAS$$

Dove:

Variabile	Descrizione
K_P	guadagno proporzionale (adimensionale) $K_p = K_c$ adimensionale
K_i	guadagno integrale (secondi $^{-1}$) La relazione tra K_i (guadagno integrale) e T_i (tempo integrale), è: $K_i = \frac{K_C}{60T_i}$
K_d	guadagno derivativo (secondi) La relazione tra K_d (guadagno derivativo) e T_d (tempo derivativo), è: $K_d = K_c (T_d) 60$
K_C	guadagno del controllore (adimensionale)
T_i	tempo integrale (minuti/ripetizione)
T_d	tempo derivativo (minuti)
SP	setpoint
PV	variabile di processo
E	errore [(SP-PV) o (PV-SP)]
BIAS	compensazione anticipata o bias
CV	variabile di controllo
dt	tempo di aggiornamento dell'anello

Se non si desidera utilizzare un particolare termine dell'equazione PID, impostare i suoi guadagni a zero. Ad esempio, se non si desidera nessuna azione derivativa, impostare K_d o T_d a zero.

Antiaccumulo dell'integrale e trasferimento bumpless da manuale ad automatico

L'istruzione PID evita automaticamente l'accumulo dell'integrale evitando che questo accumuli ogni volta che l'uscita della variabile di controllo raggiunge il suo valore massimo o minimo, in base all'impostazione di .MAXO e .MINO. L'integrale accumulato rimane bloccato fino a quando l'uscita della variabile di controllo non scende al di sotto del limite massimo o sale al di sopra del limite minimo. A questo punto la normale accumulazione dell'integrale riprende automaticamente.

L'istruzione PID supporta due modalità manuali di controllo:

Modalità manuale di controllo	Descrizione
Manuale software (.SWM)	conosciuta anche come modalità di impostazione uscita consente all'utente di impostare la % dell'uscita dal software Il valore di uscita impostato (.SO) viene utilizzato come uscita dell'anello. Questo valore in genere viene immesso da un operatore tramite un dispositivo di interfaccia operatore.
Manuale (.MO)	considera il valore di tieback come un ingresso e regola le sue variabili interne in modo da generare lo stesso valore all'uscita L'ingresso tieback per l'istruzione PID è in scala dallo 0 al 100 % con riferimento ai valori di .MINTIE e .MAXTIE ed è utilizzato come uscita dell'anello. L'ingresso tieback in genere deriva dall'uscita di una stazione hardware manuale/automatica che aggira l'uscita dal controllore. **Nota:** se si impostano entrambi i bit di modalità, prevale la modalità Manuale.

L'istruzione PID consente inoltre trasferimenti graduali automatici dalla modalità manuale software alla modalità Auto o dalla modalità manuale alla modalità Auto. L'istruzione PID calcola nuovamente il valore accumulato dell'integrale richiesto per fare in modo che l'uscita della variabile di controllo insegua o il valore impostato dell'uscita (.SO) nella modalità manuale software o l'ingresso tieback nella modalità manuale. In questo modo, quando l'anello passa alla modalità Auto, l'uscita della variabile di controllo inizia dal valore di uscita impostato o di tieback senza che si verifichino "picchi" nel valore dell'uscita.

L'istruzione PID è inoltre in grado di fornire automaticamente un trasferimento bumpless da Manuale ad Auto anche senza l'uso del controllo integrale (cioè $K_i = 0$). In questo caso l'istruzione modifica il termine – BIAS per far sì che l'uscite della variabile di controllo insegua o i valori di uscita impostata o di tieback. Quando viene ripristinato il controllo automatico, il termine .BIAS mantiene il suo ultimo valore. È possibile disabilitare il calcolo del .BIAS impostando il bit .NOBC della struttura dati PID. Considerare che se si imposta il bit .NOBC su vero, l'istruzione PID non fornisce più un trasferimento bumpless da Manuale ad Auto quando non si usa il controllo integrale.

Temporizzazione dell'istruzione PID

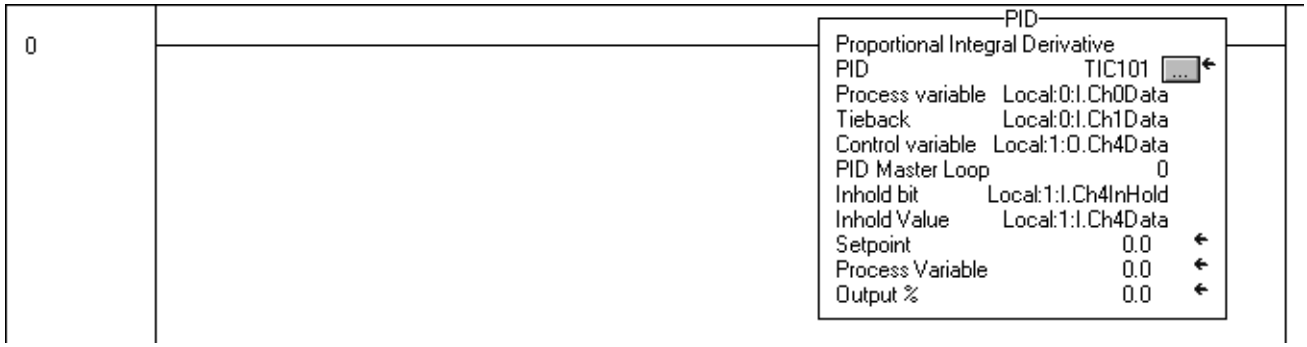
L'istruzione PID ed il campionamento della variabile di processo devono essere aggiornati con frequenza periodica. Questo tempo di aggiornamento è in stretta relazione con il processo fisico che si controlla. Nel caso di anelli molto lenti, ad esempio anelli di temperatura, un tempo di aggiornamento di una volta al secondo o più, è in genere sufficiente ad ottenere un buon controllo. Anelli più veloci, come gli anelli di pressione o di flusso, possono richiedere un

tempo di aggiornamento di una volta ogni 250 millisecondi. Solo in casi molto rari, come nel caso del controllo della tensione di un svolgitore di bobine, è necessario un aggiornamento dell'anello ogni 10 millisecondi o più veloce.

Poiché l'istruzione PID utilizza una base tempo per i suoi calcoli, è necessario sincronizzare l'esecuzione di questa istruzione con il campionamento della variabile di processo (PV).

Il modo più semplice per eseguire l'istruzione PID è quello di inserire l'istruzione PID in un task periodico. Configurare il tempo di aggiornamento dell'anello (.UPD) uguale alla frequenza del task periodico ed accertarsi che l'istruzione PID venga eseguita ad ogni scansione del task periodico.

Ladder



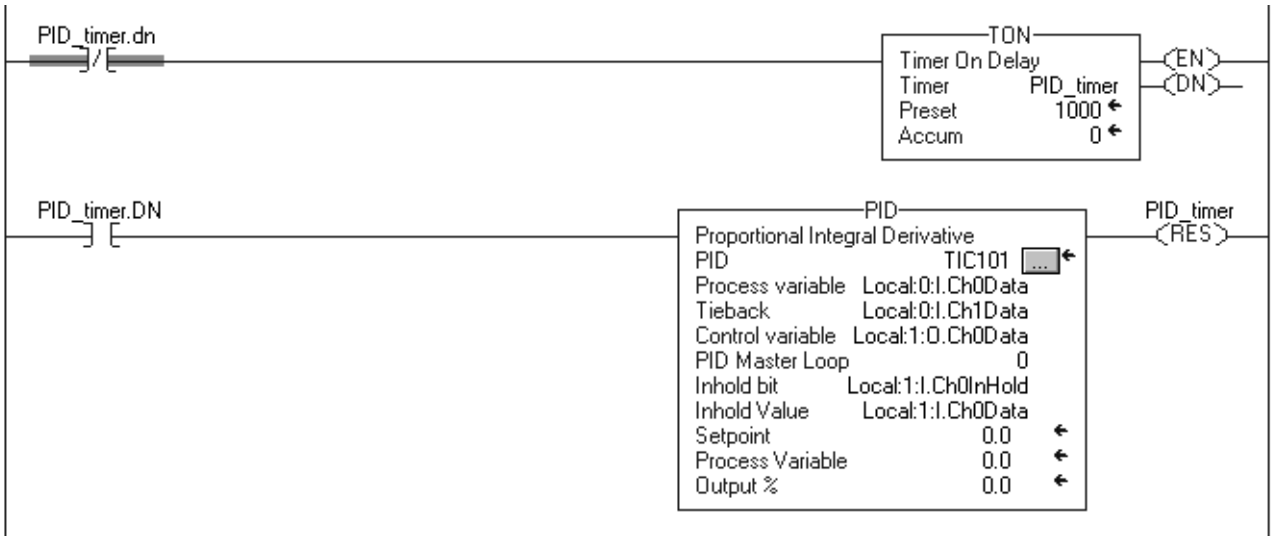
Testo strutturato

```
PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data,
    Local:1:O.Ch4Data,0,Local:1:I.Ch4InHold,
    Local:1:I.Ch4Data);
```

Quando si utilizza un task periodico, accertarsi che l'ingresso analogico utilizzato per la variabile di processo venga aggiornato nel processore con una frequenza significativamente più veloce di quella del task periodico. Idealmente, la variabile di processo dovrebbe essere inviata al processore con una velocità almeno 5–10 volte superiore a quella del task periodico. Ciò consente di ridurre la differenza di tempo tra i campioni effettivi della variabile di processo e l'esecuzione dell'anello PID. Per esempio, se l'anello PID si trova in un task periodico di 250 millisecondi, utilizzare un tempo di aggiornamento dell'anello di 250 millisecondi (.UPD = .25) e configurare il modulo di ingresso analogico in modo che produca dati almeno ogni 25–50 msec circa.

Un altro metodo, ma meno preciso, di eseguire un'istruzione PID è quello di inserire l'istruzione in un task continuo ed utilizzare il bit di fine di un timer per attivare l'esecuzione dell'istruzione PID.

Ladder



Testo strutturato

```
PID_timer.pre := 1000

TONR(PID_timer);

IF PID_timer.DN THEN

     PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data,
     Local:1:O.Ch0Data,0,Local:1:I.Ch0InHold,
     Local:1:I.Ch0Data);

END_IF;
```

Con questo metodo, il tempo di aggiornamento dell'anello dell'istruzione PID deve essere posto uguale al valore preimpostato del timer. Come nel caso dell'utilizzo di un task periodico, è necessario impostare il modulo di ingresso analogico in modo che produca la variabile di processo ad una velocità molto più rapida del tempo di aggiornamento dell'anello. Il metodo mediante timer deve essere utilizzato solo per anelli con tempi di aggiornamento molto più lunghi rispetto al caso peggiore del tempo di esecuzione del task continuo.

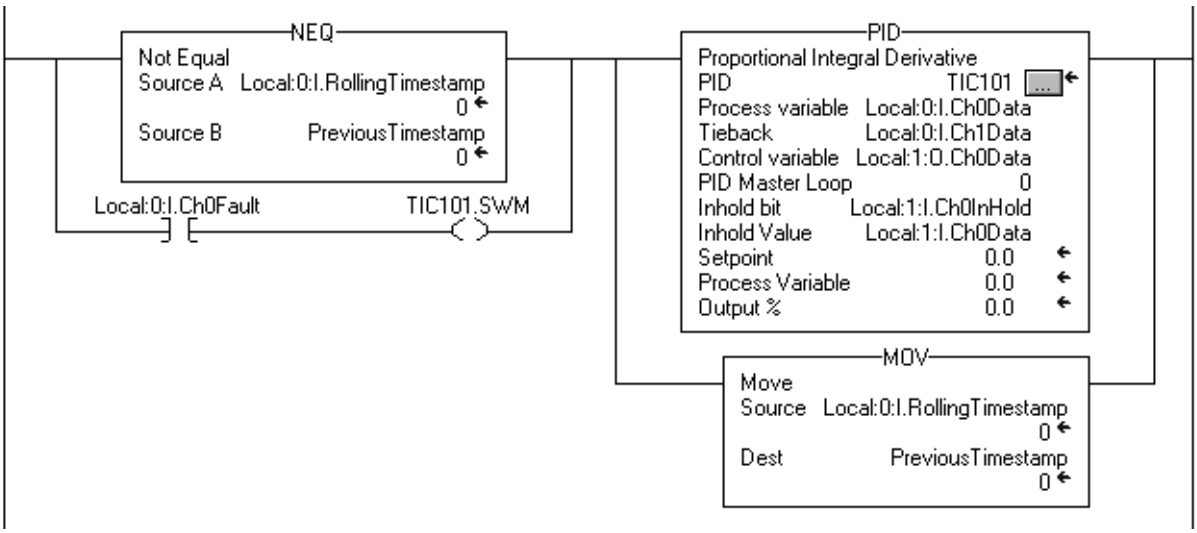
Il modo più preciso per l'esecuzione di un'istruzione PID è quello di usare la funzione di campionamento in tempo reale (RTS) dei moduli di ingresso analogico 1756. Il modulo di ingresso analogico campiona i suoi ingressi ad una velocità di campionamento in tempo reale impostata dall'utente al momento della configurazione del modulo. Alla scadenza del periodo di campionamento in tempo reale, il modulo aggiorna i suoi ingressi e il registratore cronologico ciclico (rappresentato dal membro .RollingTimestamp della struttura dati dell'ingresso analogico) prodotto dal modulo.

Il marcatore di tempo conta da 0 a 32 767 millisecondi. Controllare la registrazione cronologica. Quando cambia, significa che è stato ricevuto un nuovo campione della variabile di processo. Ogni volta che il contatore cambia, eseguire l'istruzione PID una volta. Dato che il campione della variabile di processo dipende dal modulo di ingresso analogico, il tempo del campione di ingresso è molto preciso ed il tempo di aggiornamento dell'anello utilizzato dall'istruzione PID deve essere impostato uguale al tempo dell'RTS del modulo di ingresso analogico.

Per assicurarsi di non perdere alcun campione della variabile di processo, eseguire la logica ad una velocità maggiore del tempo dell'RTS. Per esempio, se il tempo RTS è di 250 msec., si potrebbe inserire la logica PID in un task periodico che viene eseguito ogni 100 msec. per fare in modo che non si perda alcun campione. È possibile, inoltre, inserire la logica PID in un task continuo, purché ci si assicuri che la logica venga aggiornata più frequentemente di una volta ogni 250 millisecondi.

Di seguito viene fornito un esempio del metodo di esecuzione mediante RTS. L'esecuzione dell'istruzione PID dipende dalla ricezione di nuovi dati di ingresso analogici. Se il modulo di ingresso analogico va in errore o viene rimosso, il controllore interrompe la ricezione del tempo dal marcatore e l'anello PID si interrompe. È necessario monitorare il bit di stato dell'ingresso analogico della variabile di processo e in caso di stato negativo, forzare l'anello nella modalità Manuale software ed eseguire l'anello ad ogni scansione. Ciò consente sempre all'operatore di cambiare manualmente l'uscita dell'anello PID.

Ladder



Testo strutturato

```
IF (Local:0:I.Ch0Fault) THEN
     TIC101.SWM [:=] 1;
ELSE
     TIC101.SWM := 0;
END_IF;

IF (Local:0:I.RollingTimestamp<>PreviousTimestamp) OR
     (Local:0:I.Ch0Fault) THEN

     PreviousTimestamp := Local:0:I.RollingTimestamp;

     PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data,
     Local:1:O.Ch0Data,0,Local:1:I.Ch0InHold,
     Local:1:I.Ch0Data);

END_IF;
```

Riavviamento bumpless

L'istruzione PID può interagire con i moduli di uscita analogici 1756 per supportare un riavviamento senza brusche variazioni (o bumpless) quando il controllore passa dalla modalità Program alla modalità Run oppure quando il controllore viene acceso.

Se un modulo di uscita analogico 1756 perde la comunicazione con il controllore o rileva che il controllore è in modalità Program, il modulo d'uscita analogico imposta le sue uscite alla condizione di errore specificata al momento della configurazione del modulo. Quindi, quando il controllore ritorna in modalità di esecuzione oppure ristabilisce la comunicazione con il modulo d'uscita analogico, utilizzando i parametri del Bit Inhold e del Valore Inhold sull'istruzione PID, è possibile far sì che l'istruzione PID riporti automaticamente la propria uscita della variabile di controllo uguale all'uscita analogica.

Per impostare un riavviamento bumpless:

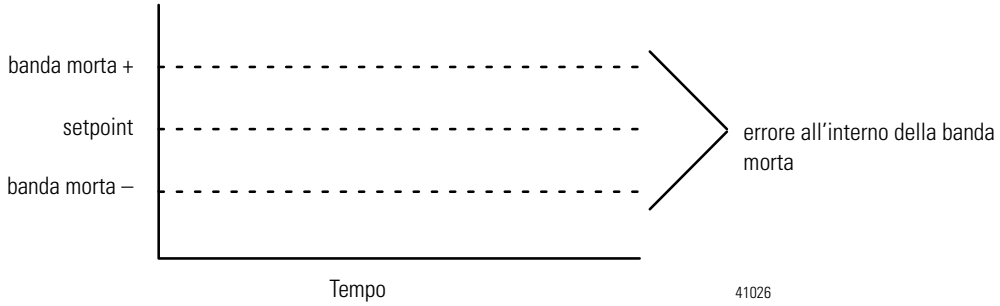
Procedere in questo modo	Dettagli
Configurare il canale del modulo di uscita analogico 1756 che riceve la variabile di controllo dall'istruzione PID	Selezionare la casella di controllo "hold for initialization" nella pagina delle proprietà dello specifico canale del modulo. In tal modo si dice al modulo di uscita analogico che quando il controllore ritorna alla modalità Run, oppure quando ristabilisce la comunicazione con il modulo, esso deve mantenere l'uscita analogica sul suo valore corrente fino a quando il valore inviato dal controllore non corrisponde (con una differenza massima dello 0,1 %) con il valore corrente utilizzato dal canale di uscita. L'uscita del controllore passerà al valore di uscita correntemente mantenuto utilizzando il termine .BIAS. Questo passaggio è simile al trasferimento bumpless ad Auto.
Inserire nell'istruzione PID il tag Inhold bit e il tag Inhold Value	Il modulo di uscita analogico 1756 ritorna due valori per ciascun canale nella sua struttura dati di ingresso. Il bit di stato InHold (.Ch2InHold, per esempio), se è vero, indica che il canale di uscita analogico sta mantenendo il suo valore. Il valore Data readback (.Ch2Data, per esempio) mostra il valore corrente dell'uscita in unità ingegneristiche. Inserire il tag del bit di stato InHold come parametro Inhold bit dell'istruzione PID. Inserire il tag del valore Data readback come parametro di Inhold Value. Se Inhold bit diventa vero, l'istruzione PID sposta Inhold Value nell'uscita della variabile di controllo e viene inizializzata di nuovo per supportare un riavviamento graduale con quel valore. Quando il modulo di uscita analogico riceve questo valore dal controllore, esso disabilita il bit di stato InHold, consentendo all'istruzione PID di iniziare il controllo normalmente.

Attenuazione derivata

Il calcolo della derivata è migliorato da un filtro di attenuazione della derivata. Questo filtro digitale passa-basso di primo livello contribuisce a ridurre i grandi picchi della derivata provocati dai disturbi nella variabile di processo. Questa attenuazione diventa più sensibile con valori del guadagno derivativo più grandi. Se il processo richiede valori di guadagno derivativo molto grandi, è possibile disabilitare l'attenuazione della derivata (K_d > 10, per esempio). Per disabilitare l'attenuazione derivativa, selezionare l'opzione "No derivative smoothing" della scheda Configurazione oppure impostare il bit .NDF della struttura PID.

Impostazione della banda morta

La banda morta regolabile permette di selezionare una gamma di errore, al di sopra e al di sotto del setpoint, in cui l'uscita non cambia fino a quando l'errore rimane entro i limiti della gamma stessa. Questa banda morta consente di controllare di quanto la variabile di processo ed il setpoint possono discostarsi senza dover cambiare l'uscita. Inoltre, la banda morta contribuisce a ridurre l'usura ed il logorio del dispositivo di controllo finale.



L'attraversamento dello zero è il controllo della banda morta che permette all'istruzione di utilizzare l'errore per scopi di calcolo dal momento in cui la variabile di processo attraversa la banda morta fino all'attraversamento del setpoint. Una volta che la variabile di processo attraversa il setpoint (l'errore attraversa lo zero e cambia di segno) e fino a quando la variabile di processo rimane nella banda morta, l'uscita non cambia.

La banda morta si estende al di sopra e al di sotto del setpoint in base ad un valore specificato dall'utente. Per inibire la banda morta, inserire zero. La banda morta ha le stesse unità in scala del setpoint. È possibile utilizzare la banda morta senza l'attraversamento dello zero selezionando nella scheda Configurazione l'opzione "no zero crossing for deadband" oppure impostando il bit .NOZC nella struttura PID.

Se si utilizza la banda morta, la variabile di controllo deve essere REAL altrimenti essa verrà forzata a 0 se l'errore è all'interno della banda morta.

Uso del limite dell'uscita

È possibile impostare un limite di uscita (percentuale dell'uscita) sul controllo dell'uscita. Quando l'istruzione rileva che l'uscita ha raggiunto un limite, essa imposta un bit di allarme impedendo all'uscita di superare il limite inferiore o superiore.

Compensazione anticipata o bias dell'uscita

È possibile anticipare un disturbo dal sistema inserendo il valore .BIAS nel valore di compensazione anticipata/bias dell'istruzione PID.

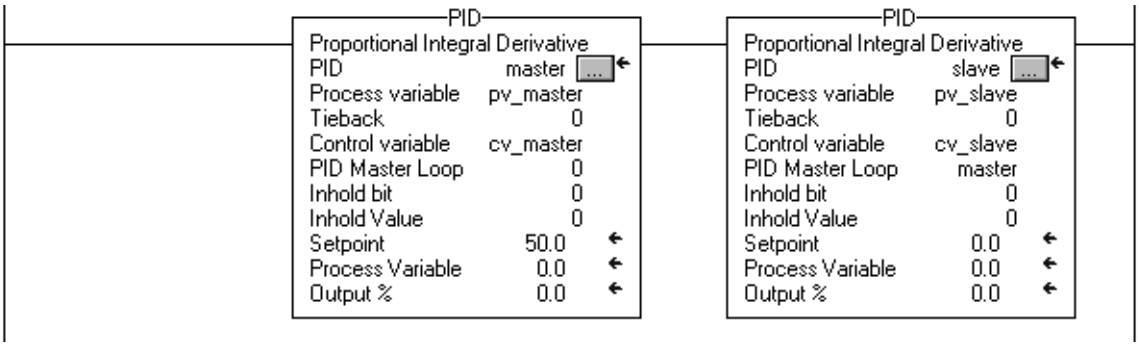
Il valore della compensazione anticipata rappresenta un disturbo immesso nell'istruzione PID prima che il disturbo possa cambiare la variabile di processo. La compensazione anticipata viene spesso utilizzata per controllare i processi con ritardo di propagazione. Per esempio, un valore della compensazione anticipata che rappresenta "l'acqua fredda versata in una miscela calda" potrebbe fare aumentare il valore dell'uscita molto più velocemente piuttosto che aspettare che la variabile di processo cambi in conseguenza della nuova condizione creatasi.

Un valore di bias in genere viene utilizzato quando non si usa l'integrale del controllo. In questo caso, il valore bias può essere regolato in modo da mantenere l'uscita entro la gamma richiesta e tenere la variabile di processo vicino al setpoint.

Anelli in cascata

L'istruzione PID pone in cascata due anelli assegnando l'uscita in percentuale dell'anello master al setpoint dell'anello slave. L'anello slave converte automaticamente l'uscita dell'anello master nelle unità ingegneristiche corrette per il setpoint dell'anello slave, in base ai valori .MAXS e .MINS dell'anello slave.

Ladder



Testo strutturato

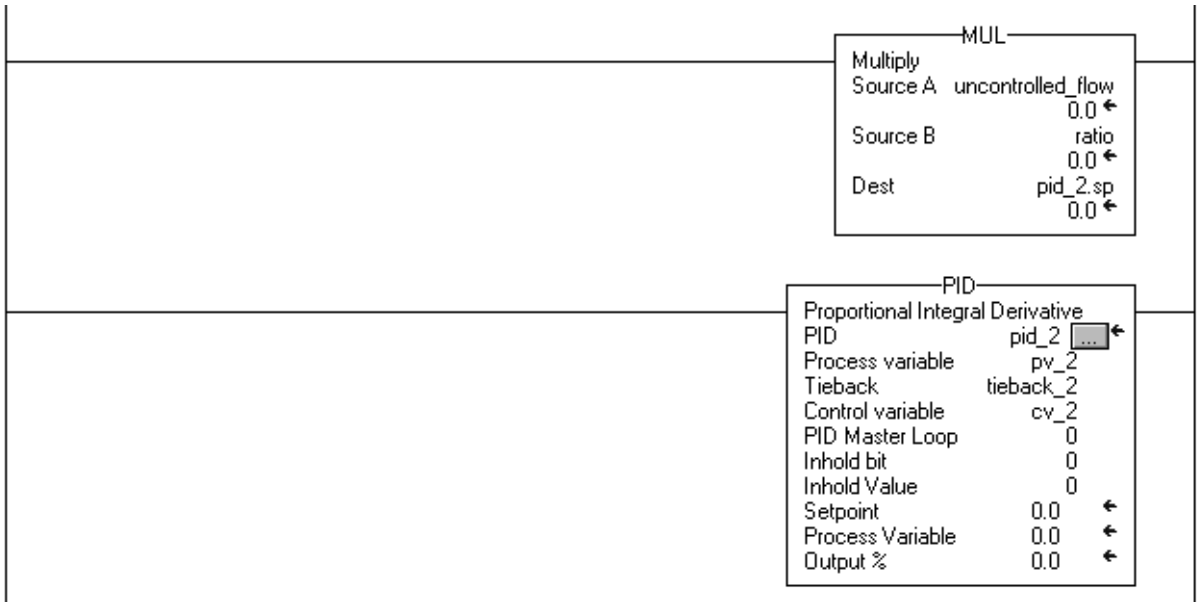
```
PID(master,pv_master,0,cv_master,0,0,0);

PID (slave,pv_slave,0,cv_slave,master,0,0);
```

Controllo di un rapporto

È possibile stabilire un rapporto tra due valori utilizzando questi parametri:

- valore non controllato
- valore controllato (il risultante setpoint che deve essere utilizzato dall'istruzione PID)
- rapporto tra questi due valori

Ladder



Testo strutturato

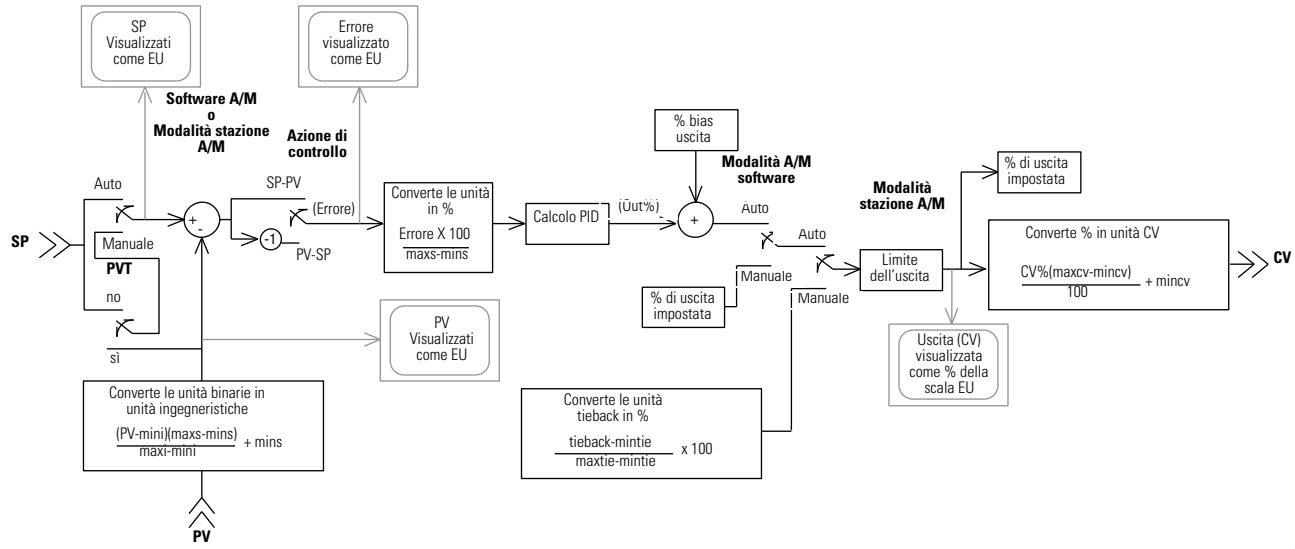
```
pid_2.sp := uncontrolled_flow * ratio

PID(pid_2,pv_2,tieback_2,cv_2,0,0,0);
```

Per questo parametro di moltiplicazione	Inserire questo valore
Destination	valore controllato
Source A	valore non controllato
Source B	rapporto

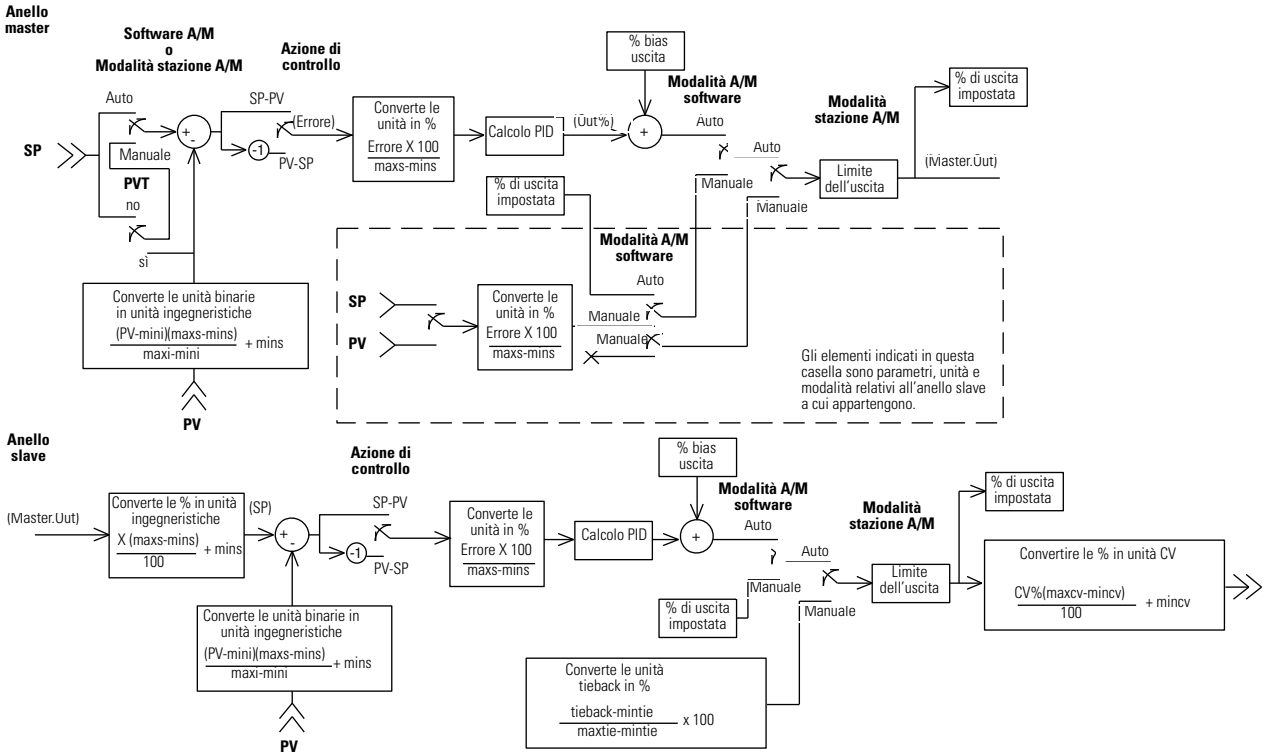
Teoria PID

Nelle figure che seguono è illustrato il flusso di processo di un'istruzione PID.

Processo PID



Processo PID con anelli master/slave



Note:

Istruzioni trigonometriche
(SIN, COS, TAN, ASN, ASIN, ACS, ACOS, ATN, ATAN)

Introduzione

Le istruzioni trigonometriche eseguono il calcolo delle funzioni trigonometriche.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
Calcolare il seno di un valore.	SIN	ladder testo strutturato blocco funzione	512
Calcolare il coseno di un valore.	COS	ladder testo strutturato blocco funzione	515
Calcolare la tangente di un valore.	TAN	ladder testo strutturato blocco funzione	518
Calcolare l'arcoseno di un valore.	ASN ASIN(1)	ladder testo strutturato blocco funzione	521
Calcolare l'arcocoseno di un valore.	ACS ACOS(1)	ladder testo strutturato blocco funzione	524
Calcolare l'arcotangente di un valore.	ATN ATAN(1)	ladder testo strutturato blocco funzione	527

(1) Solo testo strutturato.

È possibile usare tipi di dati diversi, ma si potrebbe perdere in precisione ed in arrotondamento ed inoltre l'istruzione potrebbe impiegare più tempo per essere eseguita. Controllare il bit di stato overflow (S:V) per verificare se il risultato è stato troncato.

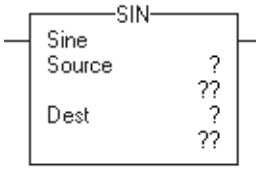
Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimali, in genere DINT o REAL.

Sine (SIN)

L'istruzione SIN calcola il seno del valore Source (in radianti) e memorizza il risultato in Destination.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcola il seno di questo valore
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := SIN(source);
```

Usare SIN come funzione. L'istruzione SIN calcola il seno di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione

Operando	Tipo	Formato	Descrizione
SIN tag	FBD_MATH_ADVANCED	struttura	struttura SIN

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione matematica Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Il valore Source deve essere maggiore o uguale a –205887,4 ($-2\pi x2^{15}$) ed inferiore o uguale a 205887,4 ($2\pi x2^{15}$). Il valore risultante di Destination è sempre maggiore o uguale a –1 e minore o uguale a 1.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

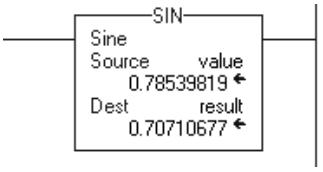
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore calcola il seno di Source ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcola il seno di value ed inserisce il risultato in result.

Ladder



Testo strutturato

```
result := SIN(value);
```

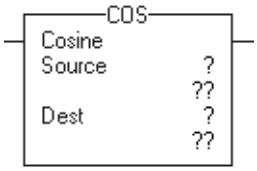
Blocco funzione

Cosine (COS)

L'istruzione COS calcola il coseno del valore Source (in radianti) e memorizza il risultato in Destination.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcola il coseno di questo valore
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

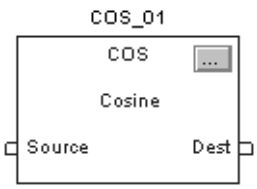
Testo strutturato

```
dest := COS(source);
```

Usare COS come funzione. L'istruzione COS calcola il coseno di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
COS tag	FBD_MATH_ADVANCED	struttura	struttura COS

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione matematica Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Il valore Source deve essere maggiore o uguale a –205887,4 ($-2\pi x2^{15}$) ed inferiore o uguale a 205887,4 ($2\pi x2^{15}$). Il valore risultante di Destination è sempre maggiore o uguale a –1 e minore o uguale a 1.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

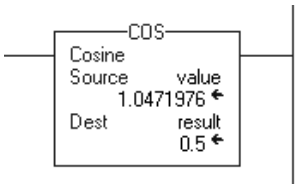
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore calcola il coseno di Source ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcola il coseno di value ed inserisce il risultato in result.

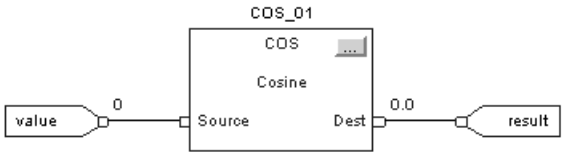
Ladder



Testo strutturato

```
result := COS(value);
```

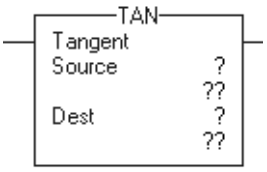
Blocco funzione



Tangent (TAN)

L'istruzione TAN calcola la tangente del valore Source (in radianti) e memorizza il risultato in Destination.

Operandi:



Ladder

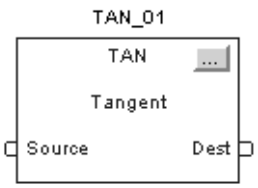
Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcola la tangente di questo valore
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

```
dest := TAN(source);
```

Testo strutturato

Usare TAN come funzione. L'istruzione TAN calcola la tangente di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
TAN tag	FBD_MATH_ADVANCED	struttura	struttura TAN

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione matematica Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Il valore Source deve essere maggiore o uguale a $-102943,7$ ($-2\pi x2^{14}$) e minore o uguale a $102943,7$ ($2\pi x2^{14}$).

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

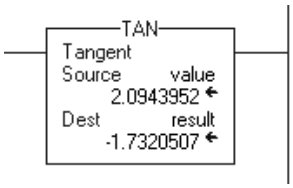
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore calcola la tangente di Source ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcola la tangente di value ed inserisce il risultato in result.

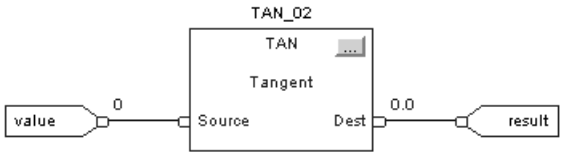
Ladder



Testo strutturato

```
result := TAN(value);
```

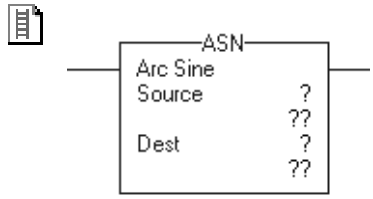
Blocco funzione



Arc Sine (ASN)

L'istruzione ASN calcola l'arcoseno del valore Source e memorizza il risultato in Destination (in radianti).

Operandi:



Ladder

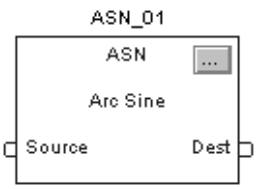
Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcola l'arcoseno di questo valore
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := ASIN(source);
```

Usare ASIN come funzione. L'istruzione ASIN calcola l'arcoseno di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
ASN tag	FBD_MATH_ADVANCED	struttura	struttura ASN

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione matematica Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Il valore Source deve essere maggiore o uguale a –1 ed inferiore o?e minore o uguale a 1. Il valore risultante di Destination è sempre maggiore o uguale a -π/2 e minore o uguale a π/2 (dove π = 3,141593).

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

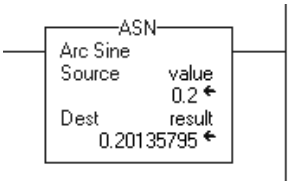
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore calcola l'arcoseno di Source ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcola l'arcoseno di value ed inserisce il risultato in result.

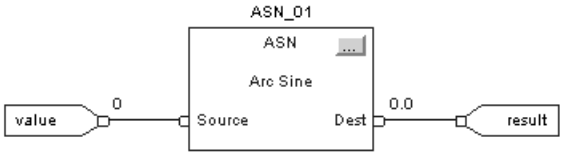
Ladder



Testo strutturato

```
result := ASIN(value);
```

Blocco funzione

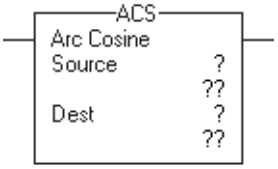


Arc Cosine (ACS)

L'istruzione ACS calcola l'arcocoseno del valore Source e memorizza il risultato in Destination (in radianti).

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcola l'arcocoseno di questo valore
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

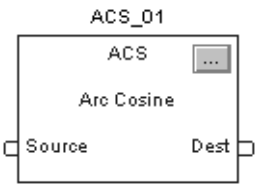
Testo strutturato

```
dest := ACOS(source);
```

Usare ACOS come funzione. L'istruzione ACOS calcola l'arcocoseno di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
ACS tag	FBD_MATH_ADVANCED	struttura	struttura ACS

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione matematica Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Il valore Source deve essere maggiore o uguale a –1 e minore o uguale a 1. Il valore risultante di Destination è sempre maggiore o uguale a 0 e minore o uguale a π (dove π = 3,141593).

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

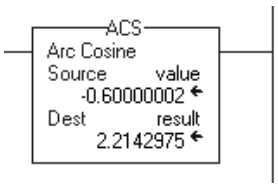
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore calcola l'arcocoseno di Source ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcola l'arcoseno di value ed inserisce il risultato in result.

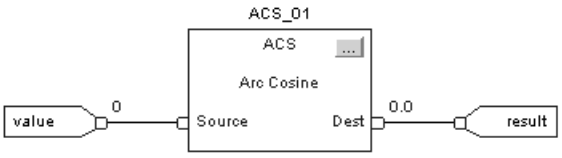
Ladder



Testo strutturato

```
result := ACOS(value);
```

Blocco funzione

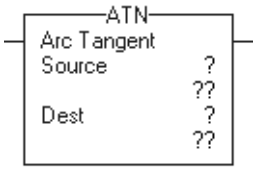


Arc Tangent (ATN)

L'istruzione ATN calcola l'arcotangente del valore Source e memorizza il risultato in Destination (in radianti).

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcola l'arcotangente di questo valore
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

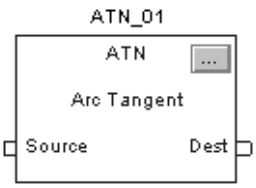
Testo strutturato

```
dest := ATAN(source);
```

Usare ATAN come funzione. L'istruzione ATAN calcola l'arcotangente di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
ATN tag	FBD_MATH_ADVANCED	struttura	struttura ATN

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione matematica Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Il valore risultante di Destination è sempre maggiore o uguale a $-\pi/2$ e minore o uguale a $\pi/2$ (dove $\pi = 3{,}141593$).

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

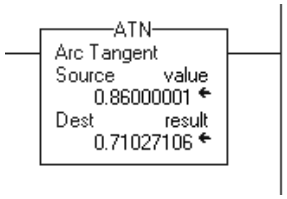
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore calcola l'arcotangente di Source ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcola l'arcotangente di value ed inserisce il risultato in result.

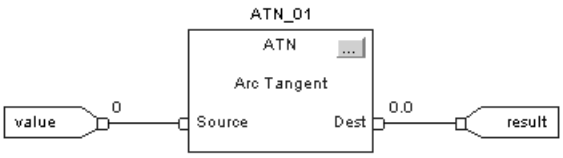
Ladder



Testo strutturato

```
result := ATAN(value);
```

Blocco funzione



Note:

Istruzioni di matematica avanzata
(LN, LOG, XPY)

Introduzione

Le istruzioni di matematica avanzata comprendono queste istruzioni:

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
Calcolare il logaritmo naturale di un valore.	LN	ladder testo strutturato blocco funzione	532
Calcolare il logaritmo in base 10 di un valore.	LOG	ladder testo strutturato blocco funzione	535
Elevare un valore alla potenza di un altro valore.	XPY	ladder testo strutturato[1] blocco funzione	538

[1] Non esiste un'istruzione equivalente in testo strutturato. Utilizzare l'operatore in un'espressione.

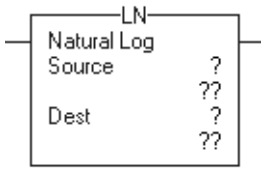
È possibile usare tipi di dati diversi, ma si potrebbe perdere in precisione ed in arrotondamento ed inoltre l'istruzione potrebbe impiegare più tempo per essere eseguita. Controllare il bit S:V per verificare se il risultato è stato troncato.

Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimali, in genere DINT o REAL.

Natural Log (LN)

L'istruzione LN calcola il logaritmo naturale di Source e memorizza il risultato in Destination.

Operandi:



Ladder

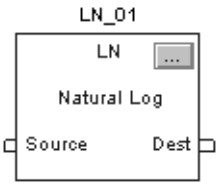
Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcola il logaritmo naturale di questo valore
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := LN(source);
```

Usare LN come funzione. Questa funzione calcola il logaritmo naturale di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
LN tag	FBD_MATH_ADVANCED	struttura	struttura LN

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione matematica. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.

Descrizione: Il valore Source deve essere maggiore di zero, altrimenti viene impostato S:V (bit di stato di overflow). Il valore Destination è maggiore o uguale a –87,33655 e minore o uguale a 88,72284.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

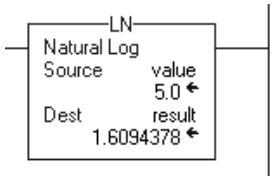
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore calcola il logaritmo naturale di Source e memorizza il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcola il logaritmo naturale di value ed inserisce il risultato in result.

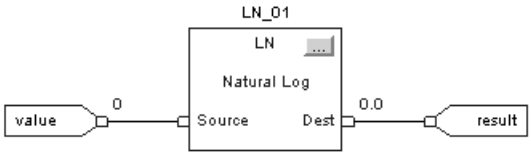
Esempio in ladder



Testo strutturato

```
result := LN(value);
```

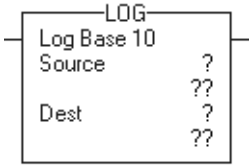
Blocco funzione



Log Base 10 (LOG)

L'istruzione LOG calcola il logaritmo in base 10 di Source e memorizza il risultato in Destination.

Operandi:



Ladder

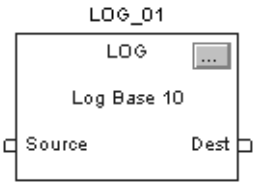
Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	calcola il logaritmo di questo valore
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := LOG(source);
```

Usare LOG come funzione. Questa funzione calcola il logaritmo di *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
LOG tag	FBD_MATH_ADVANCED	struttura	struttura LOG

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	dati di partenza dell'istruzione matematica. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Il valore Source deve essere maggiore di zero, altrimenti viene impostato S:V (bit di stato di overflow). Il valore Destination è maggiore o uguale a –37,92978 e minore o uguale a 38,53184.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

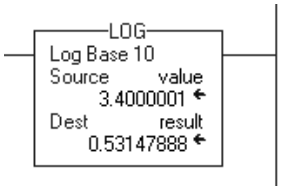
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore calcola il logaritmo di Source ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Calcola il logaritmo di value ed inserisce il risultato in result.

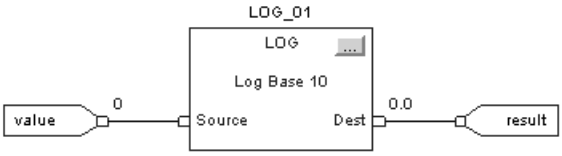
Ladder



Testo strutturato

```
result := LOG(value);
```

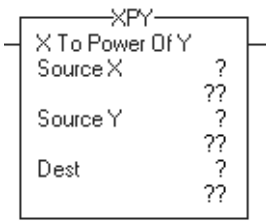
Blocco funzione



X to the Power of Y (XPY)

L'istruzione XPY eleva Source A (X) alla potenza di Source B (Y) e memorizza il risultato in Destination.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source X	SINT	immediato	valore base
	INT	tag	
	DINT		
	REAL		
Source Y	SINT	immediato	esponente
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

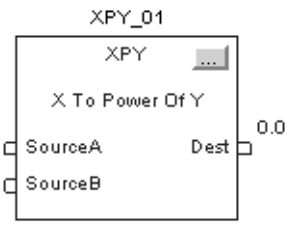
Testo strutturato

```
dest := sourceX ** sourceY;
```

Usare due segni di moltiplicazione "**" uno vicino all'altro come operatore all'interno di un'espressione. Questa espressione eleva *sourceX* alla potenza di *sourceY* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
XPY tag	FBD_MATH	struttura	struttura XPY

Struttura FBD_MATH

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source X	REAL	Valore base. Valori validi = qualsiasi numero a virgola mobile
Source Y	REAL	Esponente. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione matematica. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Se Source X è negativa, Source Y deve essere un valore intero, altrimenti si verifica un errore minore.

L'istruzione XPY utilizza questo algoritmo: *Destination* = X**Y

Il controllore calcola x^0=1 e 0^x=0.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

Si verifica un errore minore se	Tipo errore	Codice errore
Source X è negativa e Source Y non è un valore intero	4	4

Esecuzione:

Ladder

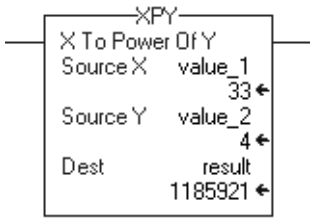
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore eleva Source X alla potenza di Source Y ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: L'istruzione XPY eleva value_1 alla potenza di value_2 ed inserisce il risultato in result.

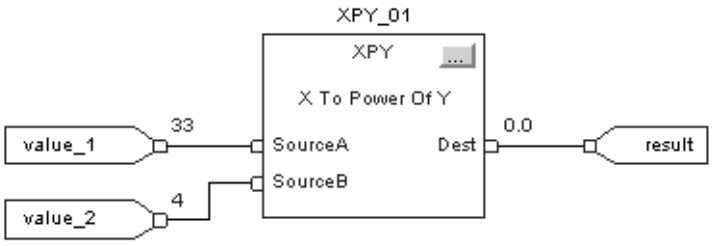
Ladder



Testo strutturato

```
result := (value_1 ** value_2);
```

Blocco funzione



Istruzioni di conversione matematica
(DEG, RAD, TOD, FRD, TRN, TRUNC)

Introduzione

Le istruzioni di conversione matematica servono a convertire i valori.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
Convertire radianti in gradi.	DEG	ladder testo strutturato blocco funzione	542
Convertire gradi in radianti.	RAD	ladder testo strutturato blocco funzione	545
Convertire un valore intero in un valore BCD.	TOD	ladder blocco funzione	548
Convertire un BCD in un valore intero.	FRD	ladder blocco funzione	551
Eliminare la parte frazionaria di un valore	TRN TRUNC(1)	ladder testo strutturato blocco funzione	553

(1) Solo testo strutturato.

È possibile usare tipi di dati diversi, ma si potrebbe perdere in precisione ed in arrotondamento ed inoltre l'istruzione potrebbe impiegare più tempo per essere eseguita. Controllare il bit S:V per verificare se il risultato è stato troncato.

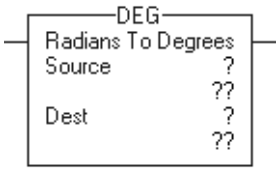
Per le istruzioni ladder, i tipi di dati in **grassetto** indicano i tipi di dati ottimali. Un'istruzione viene eseguita più velocemente e richiede meno memoria se tutti gli operandi dell'istruzione utilizzano il medesimo tipo di dati ottimali, in genere DINT o REAL.

Degrees (DEG)

L'istruzione DEG converte il valore Source (in radianti) in gradi e memorizza il risultato in Destination.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	valore da convertire in gradi
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

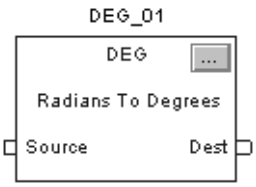
Testo strutturato

```
dest := DEG(source);
```

Usare DEG come funzione. Questa funzione converte il valore *source* in gradi e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.

Blocco funzione



Operando	Tipo	Formato	Descrizione
DEG tag	FBD_MATH_ADVANCED	struttura	struttura DEG

Struttura FBD_MATH_ADVANCED

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione di conversione. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione di conversione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: L'istruzione DEG utilizza questo algoritmo:
Source*180/π (dove π = 3,141593)

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

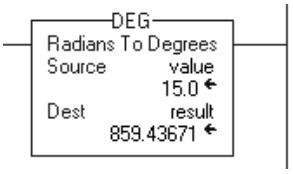
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore converte Source in gradi ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Converte value in gradi ed inserisce il risultato in result.

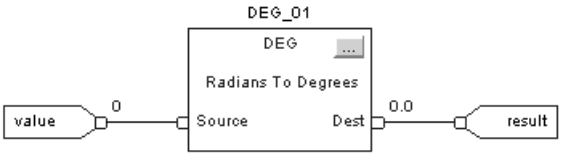
Ladder



Testo strutturato

```
result := DEG(value);
```

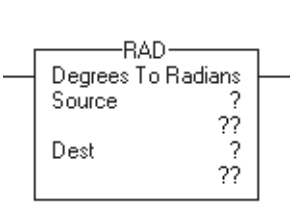
Blocco funzione



Radians (RAD)

L'istruzione RAD converte il valore Source (in gradi) in radianti e memorizza il risultato in Destination.

Operandi:



Ladder

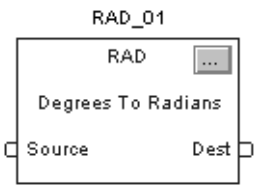
Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	valore da convertire in radianti
	INT	tag	
	DINT		
	REAL		
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := RAD(source);
```

Usare RAD come funzione. Questa funzione converte il valore *source* in radianti e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
RAD tag	FBD_MATH_ADVANCED	struttura	struttura RAD

Struttura FBD_MATH_ADVANCED

Parametro di input	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione di conversione. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	REAL	Risultato dell'istruzione di conversione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: L'istruzione RAD utilizza questo algoritmo:
Source*π/180 (dove π = 3,141593)

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

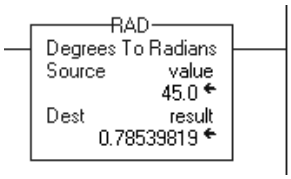
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore converte Source in radianti ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio Converte value in radianti ed inserisce il risultato in result.

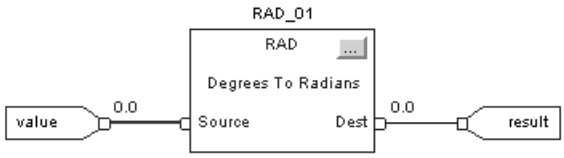
Ladder



Testo strutturato

```
result := RAD(value);
```

Blocco funzione

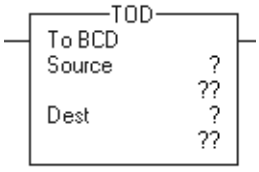


Convert to BCD (TOD)

L'istruzione TOD converte un valore decimale (0 ≤ Source ≤ 99 999 999) in un valore BCD e memorizza il risultato in Destination.

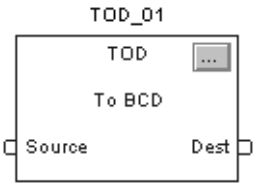
Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	valore da convertire in decimale
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Destination	SINT	tag	memorizza il risultato
	INT		
	DINT		

Blocco funzione



Operando	Tipo	Formato	Descrizione
TOD tag	FBD_CONVERT	struttura	struttura TOD

Struttura FBD_CONVERT

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	DINT	Dati di partenza dell'istruzione di conversione. Valori validi = qualsiasi numero intero
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Risultato dell'istruzione di conversione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: La sigla BCD è l'acronimo di Binary Coded Decimal (decimale codificato in binario), il sistema che esprime le singole cifre decimali (0–9) sotto forma di notazione binaria a 4 bit.

Se si inserisce un valore Source negativo, l'istruzione genera un errore minore ed azzera la Destination.

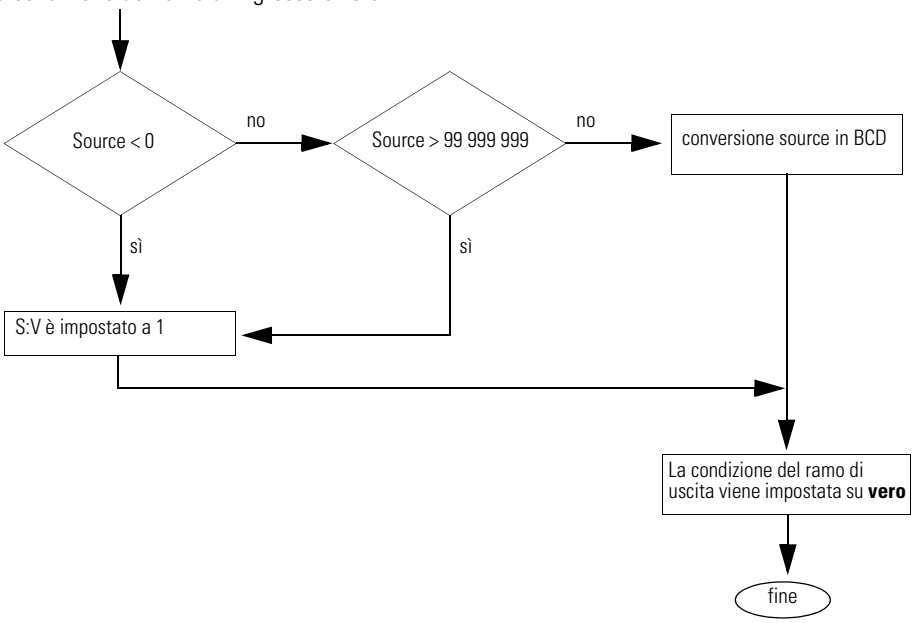
Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

Si verifica un errore minore se	Tipo errore	Codice errore
Source < 0	4	4

Esecuzione:

Ladder

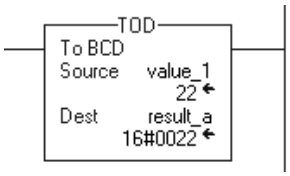
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	
se la condizione del ramo di ingresso è vera	Il controllore converte Source in BDC ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

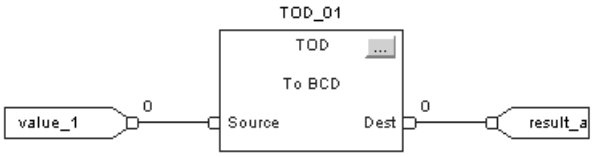
Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: L'istruzione TOD converte *value_1* in un valore BCD e inserisce il risultato in *result_a*.

Ladder



Blocco funzione

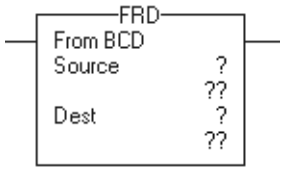


Convert to Integer (FRD)

L'istruzione FRD converte un valore BCD (Source) in un valore decimale e memorizza il risultato in Destination.

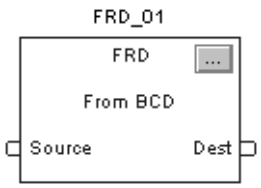
Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Source	SINT	immediato	valore da convertire in decimale
	INT	tag	
	DINT		
	Un tag SINT o INT viene convertito in un valore DINT tramite riempimento con zeri.		
Destination	SINT	tag	memorizza il risultato
	INT		
	DINT		

Blocco funzione



Operando	Tipo	Formato	Descrizione
FRD tag	FBD_CONVERT	struttura	struttura FRD

Struttura FBD_CONVERT

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	DINT	Dati di partenza dell'istruzione di conversione. Valori validi = qualsiasi numero intero
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Risultato dell'istruzione di conversione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: L'istruzione FRD converte un valore BCD (Source) in un valore decimale e memorizza il risultato in Destination.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

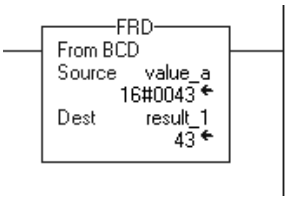
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore converte il valore Source in decimali ed inserisce il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

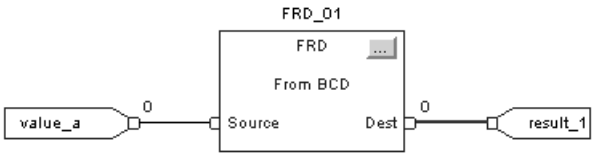
Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: L'istruzione FRD converte *value_a* in un valore decimale e inserisce il risultato in *result_1*.

Ladder



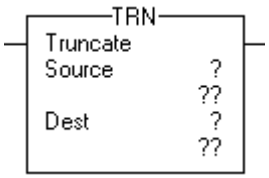
Blocco funzione



Truncate (TRN)

L'istruzione TRN elimina (tronca) la parte frazionaria di un valore (Source) e memorizza il risultato in Destination.

Operandi:



Ladder

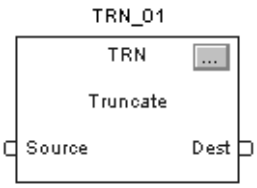
Operando	Tipo	Formato	Descrizione
Source	REAL	immediato	valore da troncare
		tag	
Destination	SINT	tag	tag dove memorizzare il risultato
	INT		
	DINT		
	REAL		

Testo strutturato

```
dest := TRUNC(source);
```

Usare TRUNC come funzione. Questa funzione tronca il valore *source* e memorizza il risultato in *dest*.

Consultare l'Appendice C, Programmazione in testo strutturato per informazioni sulla sintassi delle espressioni nel testo strutturato.



Blocco funzione

Operando	Tipo	Formato	Descrizione
TRN tag	FBD_TRUNCATE	struttura	struttura TRN

Struttura FBD_TRUNCATE

Parametro di ingresso	Tipo di dati	Descrizione
EnableIn	BOOL	Abilita gli ingressi. Se è azzerato, l'istruzione non viene eseguita e le uscite non vengono aggiornate. Il valore di default è impostato.
Source	REAL	Dati di partenza dell'istruzione di conversione. Valori validi = qualsiasi numero a virgola mobile
Parametro di uscita	**Tipo di dati**	**Descrizione**
EnableOut	BOOL	L'istruzione produce un risultato valido.
Dest	DINT	Risultato dell'istruzione di conversione. Per questa uscita vengono impostati gli indicatori di stato aritmetico.

Descrizione: Il troncamento non arrotonda il valore in quanto la parte intera rimane la stessa a prescindere dal valore della parte frazionaria.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore: nessuna

Esecuzione:

Ladder

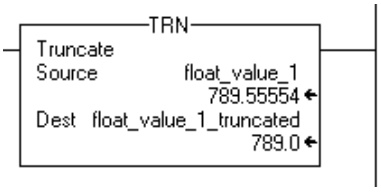
Condizione	Azione
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	Il controllore elimina la parte frazionaria di Source e memorizza il risultato in Destination. La condizione del ramo di uscita viene impostata su vero.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Blocco funzione

Condizione	Azione
durante la prescansione	Nessuna.
prima scansione dell'istruzione	Nessuna.
durante la prima esecuzione dell'istruzione	Nessuna.
EnableIn viene azzerato	EnableOut viene azzerato.
EnableIn viene impostato	L'istruzione viene eseguita. EnableOut viene impostato.
post-scansione	Nessuna.

Esempio: Elimina la parte frazionaria di *float_value_1*, lasciando invariata la parte non frazionaria e inserisce il risultato in *float_value_1_truncated*.

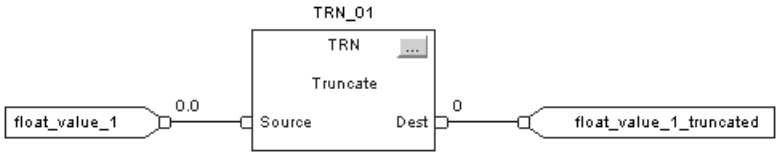
Ladder



Testo strutturato

```
float_value_1_truncated := TRUNC(float_value_1);
```

Blocco funzione



Note:

Istruzioni Porta Seriale ASCII
(ABL, ACB, ACL, AHL, ARD, ARL, AWA, AWT)

Introduzione

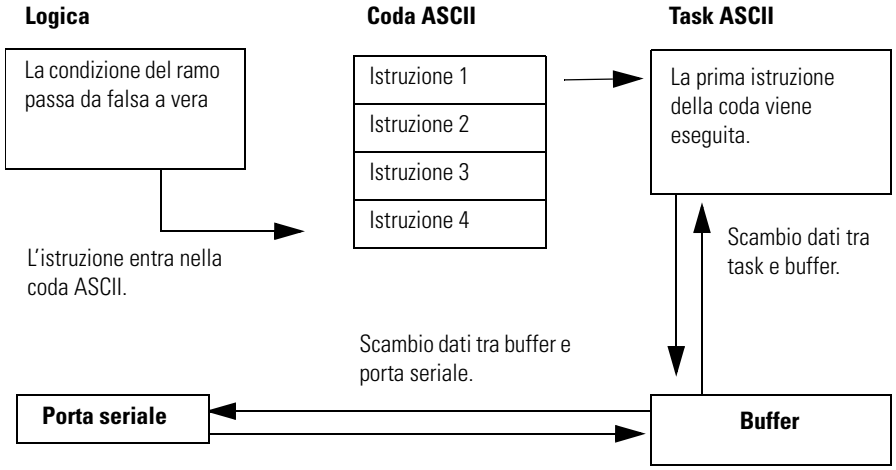
Utilizzare le istruzioni in ASCII della porta seriale per leggere e scrivere caratteri ASCII.

IMPORTANTE Per utilizzare le istruzioni porta seriale ASCII, si deve configurare la porta seriale del controllore. Vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.

Se si desidera	Per esempio	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
determinare se il buffer contiene caratteri di terminazione	controllare i dati contenenti caratteri di terminazione	ABL	ladder testo strutturato	562
contare i caratteri nel buffer	controllare il numero di caratteri richiesto prima di leggere il buffer	ACB	ladder testo strutturato	565
cancellare il buffer cancellare le istruzioni per la Porta Seriale in ASCII in corso di esecuzione o in coda	• eliminare i dati vecchi del buffer all'avviamento • sincronizzare il buffer con un dispositivo	ACL	ladder testo strutturato	567
conoscere lo stato delle linee di controllo della porta seriale attivare o disattivare il segnale DTR attivare o disattivare il segnale RTS	causare una sospensione modem	AHL	ladder testo strutturato	569
leggere un numero fisso di caratteri	leggere dati da un dispositivo che invia lo stesso numero di caratteri ad ogni trasmissione	ARD	ladder testo strutturato	573
leggere un numero di caratteri variabile fino al primo set di caratteri di terminazione (compreso)	leggere dati da un dispositivo che invia un numero di caratteri variabile ad ogni trasmissione	ARL	ladder testo strutturato	577
inviare caratteri ed includere automaticamente uno o due caratteri aggiuntivi per segnalare la fine dei dati	inviare messaggi che utilizzano sempre lo stesso carattere (o caratteri) di terminazione	AWA	ladder testo strutturato	581
inviare caratteri	inviare messaggi che utilizzano una varietà di caratteri di terminazione	AWT	ladder testo strutturato	586

Esecuzione delle istruzioni

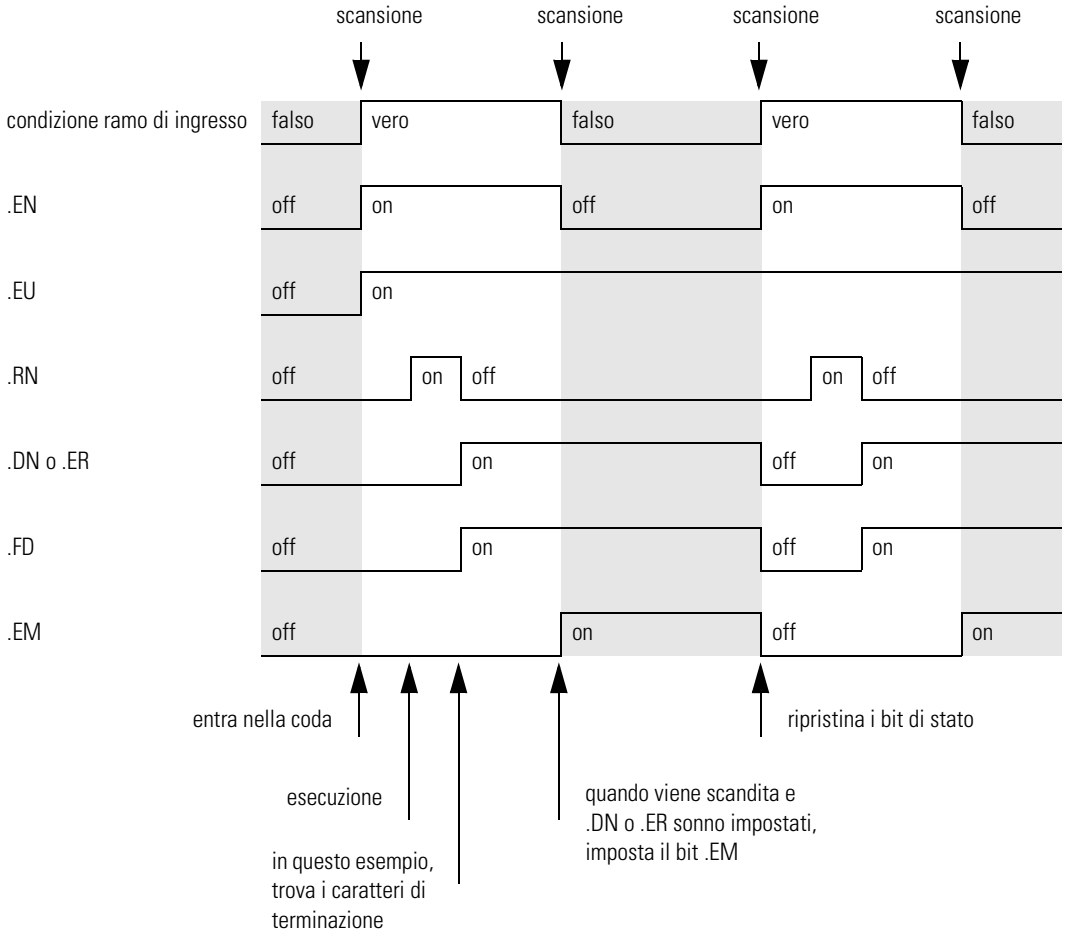
Le istruzioni porta seriale ASCII vengono eseguite in maniera asincrona rispetto alla scansione della logica:



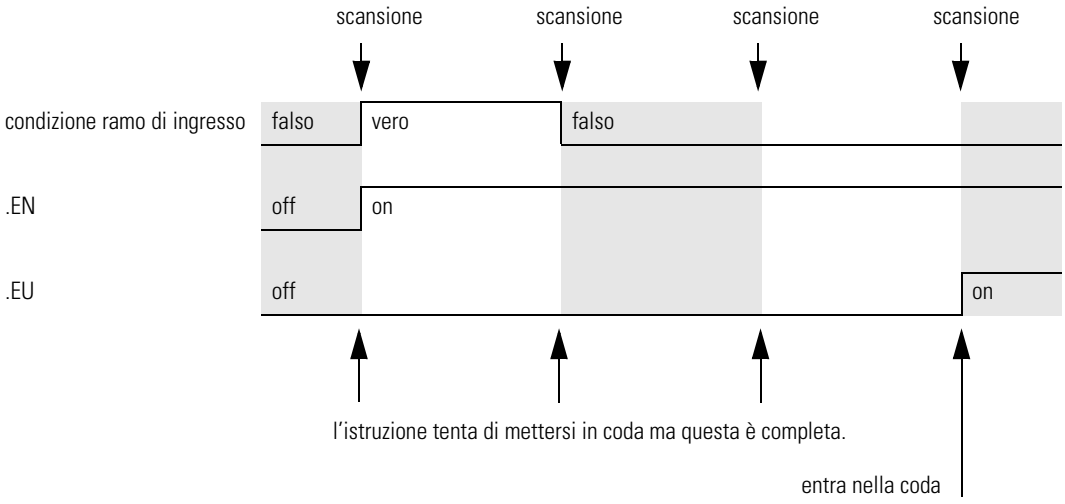
Ciascuna istruzione porta seriale ASCII (eccetto ACL) utilizza una struttura SERIAL_PORT_CONTROL per ottenere le seguenti funzioni:

- controllare l'esecuzione dell'istruzione
- fornire informazioni sullo stato dell'istruzione

Il seguente diagramma dei tempi riporta le modifiche dei bit di stato quando un'istruzione ABL controlla i caratteri di terminazione del buffer.



La coda ASCII può contenere fino a 16 istruzioni. La figura sottostante mostra come un'istruzione tenta di mettersi in coda ad ogni successiva scansione quando la coda è completa:

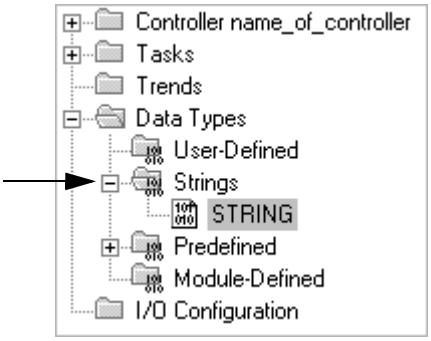


Codici di errore ASCII

Se un'istruzione porta seriale ASCII fallisce nell'esecuzione, il membro ERROR della sua struttura SERIAL_PORT_CONTROL conterrà uno dei seguenti codici di errore esadecimali:

Codice di errore esad.	Significato
16#2	Modem offline.
16#3	Segnale CTS interrotto durante la comunicazione.
16#4	Porta seriale in modalità Sistema.
16#A	Il bit .UL era impostato prima che l'istruzione fosse eseguita. Impossibile eseguire l'istruzione.
16#C	Il controllore è passato dalla modalità Run alla Modalità Program. Ciò interrompe l'esecuzione di una istruzione porta seriale ASCII ed azzera la coda.
16#D	Nella finestra di dialogo delle proprietà del controllore, scheda User Protocol, sono stati modificati i parametri di dimensione buffer e modalità eco. Ciò interrompe l'esecuzione di una istruzione porta seriale ASCII ed azzera la coda.
16#E	Istruzione ACL eseguita.
16#F	Configurazione Porta Seriale modificata da modalità Utente a modalità Sistema. Ciò interrompe l'esecuzione di una istruzione porta seriale ASCII ed azzera la coda dell'istruzione porta seriale ASCII.
16#51	Il valore LEN del tag stringa è negativo oppure maggiore della dimensione DATA del tag stringa.
16#54	Serial Port Control Length è maggiore della dimensione del buffer.
16#55	Serial Port Control Length è negativa oppure maggiore della dimensione di Source o Destination.

Tipi di dati stringa



I caratteri ASCII vengono memorizzati in tag che utilizzano un tipo di dati stringa.

- Si può utilizzare il tipo di dati STRINGA di default. Esso memorizza fino a 82 caratteri.
- Si può creare un nuovo tipo di dati stringa che memorizza più o meno caratteri.

Per creare un nuovo tipo di dati stringa, vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.

Ciascun tipo di dati stringa contiene i seguenti membri:

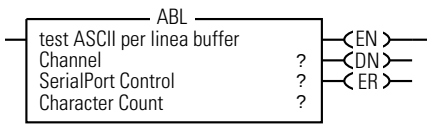
Nome	Tipo di dati	Descrizione	Note
LEN	DINT	numero di caratteri nella stringa	LEN si aggiorna automaticamente al nuovo conteggio dei caratteri ogni volta che: • si usa la finestra di dialogo Browser Stringa per inserire caratteri • si usano istruzioni che leggono, convertono o manipolano una stringa LEN mostra la lunghezza della stringa attuale. Il membro DATA può contenere caratteri vecchi addizionali che non sono inclusi nel conteggio LEN.
DATA	Matrice SINT	caratteri ASCII della stringa	• Per accedere ai caratteri della stringa, indirizzare il nome del tag. Per esempio, per accedere ai caratteri della *string_1*, inserire *string_1*. • Ciascun elemento della matrice DATA contiene un carattere. • Si possono creare nuovi tipi di dati stringa che memorizzano più o meno caratteri.

ASCII Test For Buffer Line (ABL)

L'istruzione ABL conta i caratteri nel buffer fino al primo set di caratteri di terminazione (compreso).

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Channel	DINT	immediato tag	0
Serial Port Control	SERIAL_PORT_CONTROL	tag	tag che controlla l'operazione
Character Count	DINT	immediato	0 Durante l'esecuzione visualizza il numero di caratteri nel buffer fino al primo set di caratteri di terminazione (compreso).

Testo strutturato

```
ABL(Channel
    SerialPortControl);
```

Gli operandi sono gli stessi dell'istruzione ABL in ladder. Si può accedere al valore di Character Count dal membro .POS della struttura SERIAL_PORT_CONTROL.

Struttura SERIAL_PORT_CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione è abilitata.
.EU	BOOL	Il bit di coda indica che l'istruzione è entrata nella coda ASCII.
.DN	BOOL	Il bit di fine indica quando l'istruzione è stata eseguita, però è asincrono rispetto alla scansione logica.
.RN	BOOL	Il bit di esecuzione indica che l'istruzione è in corso di esecuzione.
.EM	BOOL	Il bit di vuoto indica quando l'istruzione è stata eseguita, però è sincrono rispetto alla scansione logica.
.ER	BOOL	Il bit di errore indica quando l'istruzione fallisce (genera un errore).
.FD	BOOL	Il bit di trovato indica che l'istruzione ha trovato il carattere/i di terminazione.
.POS	DINT	La posizione determina il numero di caratteri nel buffer fino al primo set di caratteri di terminazione (compreso). L'istruzione restituisce questo valore solo dopo aver trovato il carattere/i di terminazione.
.ERROR	DINT	L'errore contiene un valore esadecimale che identifica la causa di un errore.

Descrizione L'istruzione ABL cerca il primo set di caratteri di terminazione nel buffer. Se l'istruzione trova i caratteri di terminazione:

- imposta il bit .FD
- conta i caratteri nel buffer fino al primo set di caratteri di terminazione (compreso).

La finestra di dialogo Proprietà Controllore, scheda Protocollo utente, definisce i caratteri ASCII che l'istruzione considera come caratteri di terminazione.

Per programmare un'istruzione ABL, seguire questa procedura:

1. Configurare la porta seriale del controllore per la modalità utente e definire i caratteri da utilizzare come caratteri di terminazione.

2. Questa è un'istruzione di transizione:
   - In ladder, cambia la condizione del ramo di ingresso da azzerata a impostata ogni volta che l'istruzione deve essere eseguita.
   - In testo strutturato, condiziona l' istruzione in modo che venga eseguita solo su una transizione.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	Quando la condizione del ramo passa da azzerata a impostata, l'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione conteggia i caratteri nel buffer. Il bit .EN viene impostato. I rimanenti bit di stato, tranne .UL, vengono azzerati. L'istruzione tenta di immettersi nella coda ASCII.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Verifica continuamente il buffer per i caratteri di terminazione.

Ladder

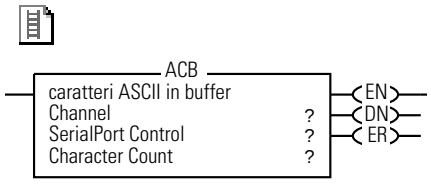
Testo strutturato

```
ABL(0,MV_line);
```

ASCII Chars in Buffer (ACB)

L'istruzione ACB conteggia i caratteri nel buffer.

Operandi:



Ladder

Operando	Tipo	Formato	Digitare
Channel	DINT	immediato tag	0
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag che controlla l'operazione
Character Count	DINT	immediato	0 Durante l'esecuzione visualizza il numero di caratteri nel buffer.

Testo strutturato

```
ACB(Channel
    SerialPortControl);
```

Gli operandi sono gli stessi dell'istruzione ACB in ladder. Tuttavia, il valore di Character Count viene indicato accedendo al membro .POS della struttura SERIAL_PORT_CONTROL invece di inserire il valore nella lista di operandi.

Struttura SERIAL_PORT_CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione è abilitata.
.EU	BOOL	Il bit di coda indica che l'istruzione è entrata nella coda ASCII.
.DN	BOOL	Il bit di fine indica quando l'istruzione è stata eseguita, però è asincrono rispetto alla scansione logica.
.RN	BOOL	Il bit di esecuzione indica che l'istruzione è in corso di esecuzione.
.EM	BOOL	Il bit di vuoto indica quando l'istruzione è stata eseguita, però è sincrono rispetto alla scansione logica.
.ER	BOOL	Il bit di errore indica quando l'istruzione fallisce (genera un errore).
.FD	BOOL	Il bit di trovato indica che l'istruzione ha trovato un carattere.
.POS	DINT	La posizione determina il numero di caratteri nel buffer fino al primo set di caratteri di terminazione (compreso).
.ERROR	DINT	L'errore contiene un valore esadecimale che identifica la causa di un errore.

Descrizione: L'istruzione ACB conteggia i caratteri nel buffer.

Per programmare un'istruzione ACB, seguire questa procedura:

1. Configurare la porta seriale del controllore per la modalità utente.
2. Questa è un'istruzione di transizione:
   - In ladder, cambia la condizione del ramo di ingresso da azzerata a impostata ogni volta che l'istruzione deve essere eseguita.
   - In testo strutturato, condiziona l' istruzione in modo che venga eseguita solo su una transizione.

Indicatori di stato aritmetico: non influenzati

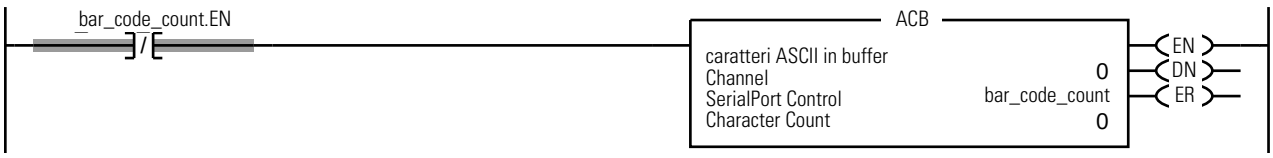
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	Quando la condizione del ramo passa da azzerata a impostata, l'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione conteggia i caratteri nel buffer. Il bit .EN viene impostato. I rimanenti bit di stato, tranne .UL, vengono azzerati. L'istruzione tenta di immettersi nella coda ASCII.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Conteggio continuo dei caratteri nel buffer.

Ladder



Testo strutturato

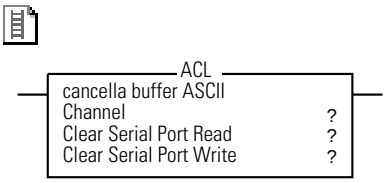
```
ACB(0,bar_code_count);
```

ASCII Clear Buffer (ACL)

L'istruzione ACL cancella immediatamente il buffer e la coda ASCI.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare
Channel	DINT	immediato tag	0
Clear Serial Port Read	BOOL	immediato tag	Per svuotare il buffer e rimuovere le istruzioni ARD e ARL dalla coda, digitare Sì.
Clear Serial Port Write	BOOL	immediato tag	Per rimuovere le istruzioni AWA e AWT dalla coda, digitare Sì.

Testo strutturato

```
ACL(Channel,
     ClearSerialPortRead,
     ClearSerialPortWrite);
```

Gli operandi sono gli stessi dell'istruzione ACL in ladder.

Descrizione: L'istruzione ACL esegue immediatamente una delle seguenti azioni:

- cancella i caratteri del buffer e le istruzioni di lettura nella coda ASCII
- cancella le istruzioni di scrittura nella coda ASCII

Per programmare un'istruzione ACL, seguire questa procedura:

1. Configurare la porta seriale del controllore:

Se l'applicAzione	Allora
usa istruzioni ARD o ARL	Selezionare la modalità Utente
non usa istruzioni ARD o ARL	Selezionare o la modalità Utente o la modalità Sistema

2. Per determinare se un'istruzione è stata eliminata dalla coda o interrotta, esaminare nell'istruzione appropriata quanto segue:
   - bit .ER viene impostato
   - il membro .ERROR è 16#E

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione azzera le istruzioni e i buffer specificati.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Quando il controllore passa in Modalità Run, azzera il buffer e la coda ASCII.

Ladder

Testo strutturato

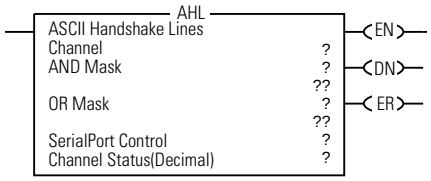
```
osri_1.InputBit := S:FS;
OSRI(osri_1);


IF (osri_1.OutputBit) THEN
     ACL(0,0,1);
END_IF;
```

ASCII Handshake Lines (AHL)

L'istruzione AHL consente di conoscere lo stato delle righe di controllo e di attivare o disattivare i segnali DTR e RTS.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare
Channel	DINT	immediato tag	0
ANDMask	DINT	immediato tag	Fare riferimento alla descrizione.
ORMask	DINT	immediato tag	
Serial Port Control	SERIAL_PORT_CONTROL	tag	tag che controlla l'operazione
Channel Status (Decimal)	DINT	immediato	0

Durante l'esecuzione, visualizza lo stato delle linee di controllo.

Per conoscere lo stato di questa linea di controllo:	Esamina questo bit:
CTS	0
RTS	1
DSR	2
DCD	3
DTR	4
Ricevuto carattere XOFF	5

Testo strutturato

```
AHL(Channel,ANDMask,ORMask,
    SerialPortControl);
```

Gli operandi sono gli stessi dell'istruzione AHL in ladder. Tuttavia, il valore Channel Status viene indicato accedendo al membro .POS della struttura SERIAL_PORT_CONTROL invece di inserire il valore nella lista di operandi.

Struttura SERIAL_PORT_CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione è abilitata.
.EU	BOOL	Il bit di coda indica che l'istruzione è entrata nella coda ASCII.
.DN	BOOL	Il bit di fine indica quando l'istruzione è stata eseguita, però è asincrono rispetto alla scansione logica.
.RN	BOOL	Il bit di esecuzione indica che l'istruzione è in corso di esecuzione.
.EM	BOOL	Il bit di vuoto indica quando l'istruzione è stata eseguita, però è sincrono rispetto alla scansione logica.
.ER	BOOL	Il bit di errore indica quando l'istruzione fallisce (genera un errore).
.FD	BOOL	Il bit di trovato non riguarda questa istruzione.
.POS	DINT	La posizione memorizza lo stato delle linee di controllo
.ERROR	DINT	L'errore contiene un valore esadecimale che identifica la causa di un errore.

Descrizione: L'istruzione AHL consente di:

- conoscere lo stato delle linee di controllo della porta seriale
- attivare o disattivare il segnale DTR (terminale pronto)
- attivare o disattivare il segnale RTS (richiesta di invio)

Per programmare un'istruzione AHL, seguire questa procedura:

1. Configurare la porta seriale del controllore:

Se l'applicAzione	Allora
usa istruzioni ARD o ARL	Selezionare la modalità Utente
non usa le istruzioni ARD o ARL	Selezionare o la modalità Utente o la modalità Sistema

2. Utilizzare la seguente tabella per scegliere i valori corretti per gli operandi ANDMask e ORMask.

Per commutare DTR	e per commutare RTS	Inserire questo valore ANDMask	Ed inserire questo valore ORMask
off	off	3	0
	on	1	2
	invariato	1	0
on	off	2	1
	on	0	3
	invariato	0	1
invariato	off	2	0
	on	0	2
	invariato	0	0

3. Questa è un'istruzione di transizione:
   - In ladder, cambia la condizione del ramo di ingresso da azzerata a impostata ogni volta che l'istruzione deve essere eseguita.
   - In testo strutturato, condiziona l' istruzione in modo che venga eseguita solo su una transizione.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

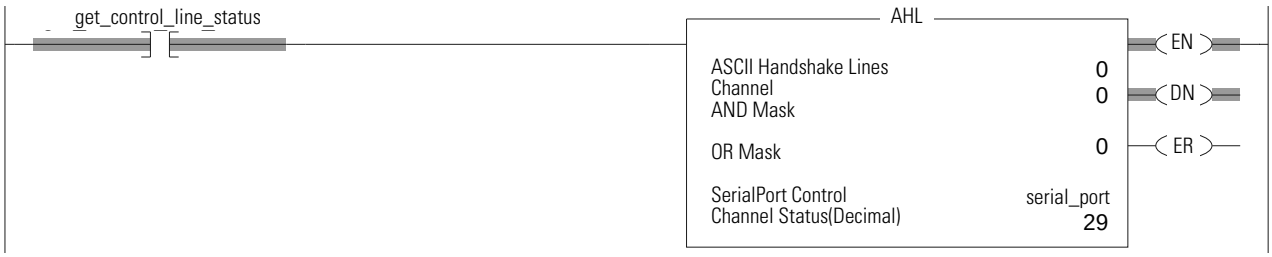
Tipo	Codice	Causa	Metodo di ripristino
4	57	L'istruzione AHL fallisce nell'esecuzione perché la porta seriale è impostata per nessun'handshaking.	Uno o l'altro: • Modificare l'impostazione di Control Line della porta seriale. • Eliminare l'istruzione AHL.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	Quando la condizione del ramo passa da azzerata a impostata, l'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione consente di conoscere lo stato delle linee di controllo e di attivare o disattivare i segnali DTR e RTS. Il bit .EN viene impostato. I rimanenti bit di stato, tranne .UL, vengono azzerati. L'istruzione tenta di immettersi nella coda ASCII.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Quando *get_control_line_status* è impostato, l'istruzione ottiene lo stato delle linee di controllo della porta seriale e lo visualizza nell'operando Channel Status. Per visualizzare lo stato di una specifica linea di controllo, monitorare il tag SerialPort Control ed espandere il membro POS.

Ladder



Testo strutturato

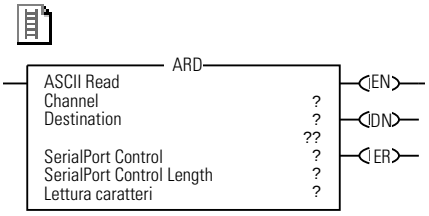
```
osri_1.InputBit := get_control_line_status;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
     AHL(0,0,0,serial_port);
END_IF;
```

ASCII Read (ARD)

L'istruzione ARD elimina i caratteri dal buffer e li memorizza nella Destination.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Channel	DINT	immediato tag	0	
Destination	**stringa** SINT INT DINT	tag	tag in cui sono spostati (lettura) i caratteri: • Per un tipo di dati stringa, inserire il nome del tag. • Per una matrice SINT, INT, o DINT, inserire il primo elemento della matrice.	• Se si vuole confrontare, convertire o manipolare i caratteri, usare un tipo di dati stringa. • I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag che controlla l'operazione	
Serial Port Control Length	DINT	immediato	numero di caratteri da spostare nella destinazione (lettura)	• La Serial Port Control Length deve essere inferiore o uguale alla dimensione della Destination. • Se si vuole impostare la Serial Port Control Length uguale alla dimensione della Destination, digitare 0.
Characters Read	DINT	immediato	0	Durante l'esecuzione, visualizza il numero dei caratteri che sono stati letti.

Testo strutturato

```
ARD(Channel,Destination,
    SerialPortControl);
```

Gli operandi sono gli stessi dell'istruzione ARD in ladder. Tuttavia, i valori Serial Port Control Length e Characters Read vengono indicati accedendo ai membri .LEN e .POS della struttura SERIAL_PORT_CONTROL invece di inserire i valori nella lista di operandi.

Struttura SERIAL_PORT_CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione è abilitata.
.EU	BOOL	Il bit di coda indica che l'istruzione è entrata nella coda ASCII.
.DN	BOOL	Il bit di fine indica quando l'istruzione è stata eseguita, però è asincrono rispetto alla scansione logica.
.RN	BOOL	Il bit di esecuzione indica che l'istruzione è in corso di esecuzione.
.EM	BOOL	Il bit di vuoto indica quando l'istruzione è stata eseguita, però è sincrono rispetto alla scansione logica.
.ER	BOOL	Il bit di errore indica quando l'istruzione fallisce (genera un errore).
.FD	BOOL	Il bit di trovato non riguarda questa istruzione.
.LEN	DINT	La lunghezza indica il numero di caratteri da spostare nella destinazione (lettura).
.POS	DINT	La posizione visualizza il numero di caratteri che sono stati letti.
.ERROR	DINT	L'errore contiene un valore esadecimale che identifica la causa di un errore.

Descrizione: L'istruzione ARD elimina il numero specificato di caratteri dal buffer e li memorizza nella Destination.

- L'istruzione ARD continua l'esecuzione fino a quando ha rimosso il numero di caratteri specificato (Serial Port Control Length).
- Mentre l'istruzione ARD è in esecuzione, non viene eseguita nessuna altra istruzione ASCII Serial Port.

Per programmare un'istruzione ARD, seguire questa procedura:

1. Configurare la porta seriale del controllore per la modalità utente.

2. Utilizzare i risultati di un'istruzione ACB per attivare l'istruzione ARD. In questo modo si evita che l'istruzione ARD trattenga la coda ASCII mentre attende il numero di caratteri richiesto.

3. Questa è un'istruzione di transizione:
   - In ladder, cambia la condizione del ramo di ingresso da azzerata a impostata ogni volta che l'istruzione deve essere eseguita.
   - In testo strutturato, condiziona l' istruzione in modo che venga eseguita solo su una transizione.

4. Per comandare un'azione successiva quando l'istruzione è stata eseguita, esaminare il bit EM.

Indicatori di stato aritmetico: non influenzati

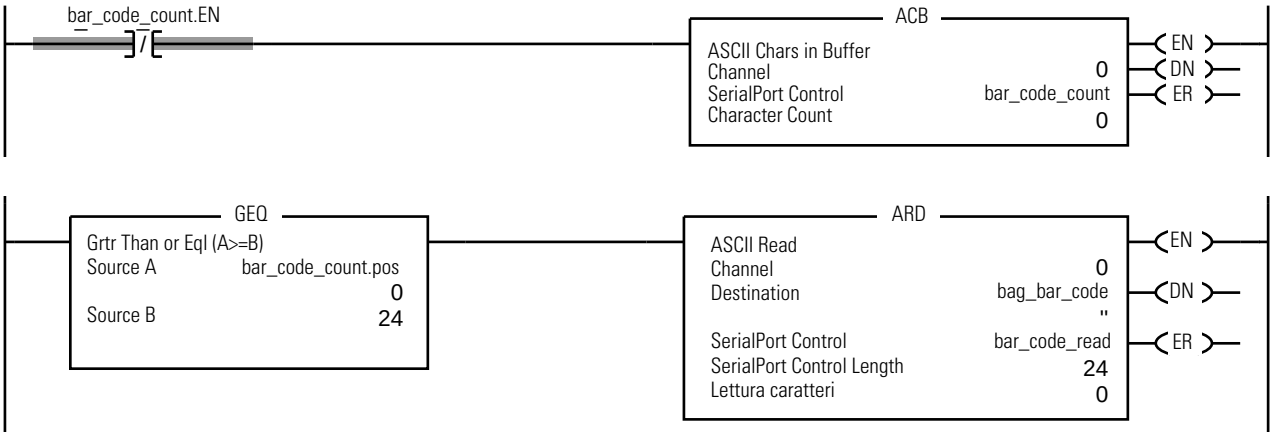
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	Quando la condizione del ramo passa da azzerata a impostata, l'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzione	L'istruzione elimina i caratteri dal buffer e li memorizza nella Destination. Il bit .EN viene impostato. I rimanenti bit di stato, tranne .UL, vengono azzerati. L'istruzione tenta di immettersi nella coda ASCII.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Un lettore di codice a barre invia i codici a barre alla porta seriale (canale 0) del controllore. Ciascun codice a barre contiene 24 caratteri. Per determinare quando il controllore riceve un codice a barre, l'istruzione ACB conteggia continuamente i caratteri nel buffer. Quando il buffer contiene almeno 24 caratteri, il controllore ha ricevuto un codice a barre. L'istruzione ARD sposta il codice a barre al membro DATA del tag *bag_bar_code*, che è una stringa.

Ladder



Testo strutturato

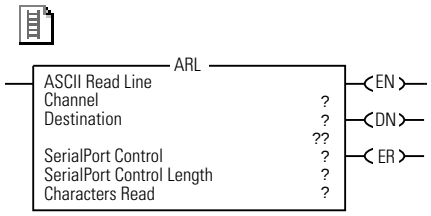
```
ACB(0,bar_code_count);
IF bar_code_count.POS >= 24 THEN
   bar_code_read.LEN := 24;
   ARD(0,bag_bar_code,bar_code_read);
END_IF;
```

ASCII Read Line (ARL)

L'istruzione ARL elimina i caratteri specificati dal buffer e li memorizza nella Destination.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Channel	DINT	immediato tag	0	
Destination	**stringa** SINT INT DINT	tag	tag in cui sono spostati (lettura) i caratteri: • Per un tipo di dati stringa, inserire il nome del tag. • Per una matrice SINT, INT, o DINT, inserire il primo elemento della matrice.	• Se si vuole confrontare, convertire o manipolare i caratteri, usare un tipo di dati stringa. • I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Serial Port Control	SERIAL_PORT_CONTROL	tag	tag che controlla l'operazione	
Serial Port Control Length	DINT	immediato	numero massimo di caratteri da leggere se non vengono trovati caratteri di terminazione	• Inserire il numero massimo di caratteri che può contenere un messaggio (cioè, quando terminare la lettura se non vengono trovati caratteri di terminazione). Per esempio, se il messaggio ha una lunghezza compresa tra 3 e 6 caratteri, inserire 6. • La Serial Port Control Length deve essere inferiore o uguale alla dimensione della Destination. • Se si vuole impostare la Serial Port Control Length uguale alla dimensione della Destination, digitare 0.
Characters Read	DINT	immediato	0	Durante l'esecuzione, visualizza il numero dei caratteri che sono stati letti.

Testo strutturato

```
ARL(Channel,Destination,
    SerialPortControl);
```

Gli operandi sono gli stessi dell'istruzione ARL in ladder. Tuttavia, i valori Serial Port Control Length e Characters Read vengono indicati accedendo ai membri .LEN e .POS della struttura SERIAL_PORT_CONTROL invece di inserire i valori nella lista di operandi.

Struttura SERIAL_PORT_CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione è abilitata.
.EU	BOOL	Il bit di coda indica che l'istruzione è entrata nella coda ASCII.
.DN	BOOL	Il bit di fine indica quando l'istruzione è stata eseguita, però è asincrono rispetto alla scansione logica.
.RN	BOOL	Il bit di esecuzione indica che l'istruzione è in corso di esecuzione.
.EM	BOOL	Il bit di vuoto indica quando l'istruzione è stata eseguita, però è sincrono rispetto alla scansione logica.
.ER	BOOL	Il bit di errore indica quando l'istruzione fallisce (genera un errore).
.FD	BOOL	Il bit di trovato non riguarda questa istruzione.
.LEN	DINT	La lunghezza indica il numero massimo di caratteri da spostare nella destinazione (cioè, quando terminare la lettura se non vengono trovati caratteri di terminazione).
.POS	DINT	La posizione visualizza il numero di caratteri che sono stati letti.
.ERROR	DINT	L'errore contiene un valore esadecimale che identifica la causa di un errore.

Descrizione: L'istruzione ARL elimina i caratteri specificati dal buffer e li memorizza nella Destination secondo le seguenti modalità:

- L'istruzione ARL continua l'esecuzione fino a quando non rimuove:
  - il primo set di caratteri di terminazione
  - numero di caratteri specificato (Serial Port Control Length)
- Mentre l'istruzione ARL è in esecuzione, non viene eseguita nessuna altra istruzione ASCII porta seriale.

Per programmare un'istruzione ARL, seguire questa procedura:

1. Configurare la porta seriale del controllore:
   a. Selezionare la modalità Utente.
   b. Definire i caratteri da utilizzare come caratteri di terminazione.
2. Utilizzare i risultati di un'istruzione ABL per attivare l'istruzione ARL. In questo modo si evita che l'istruzione ARL trattenga la coda ASCII mentre attende i caratteri di terminazione.
3. Questa è un'istruzione di transizione:
   - In ladder, cambia la condizione del ramo di ingresso da azzerata a impostata ogni volta che l'istruzione deve essere eseguita.
   - In testo strutturato, condiziona l' istruzione in modo che venga eseguita solo su una transizione.
4. Per comandare un'azione successiva quando l'istruzione è stata eseguita, esaminare il bit EM.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

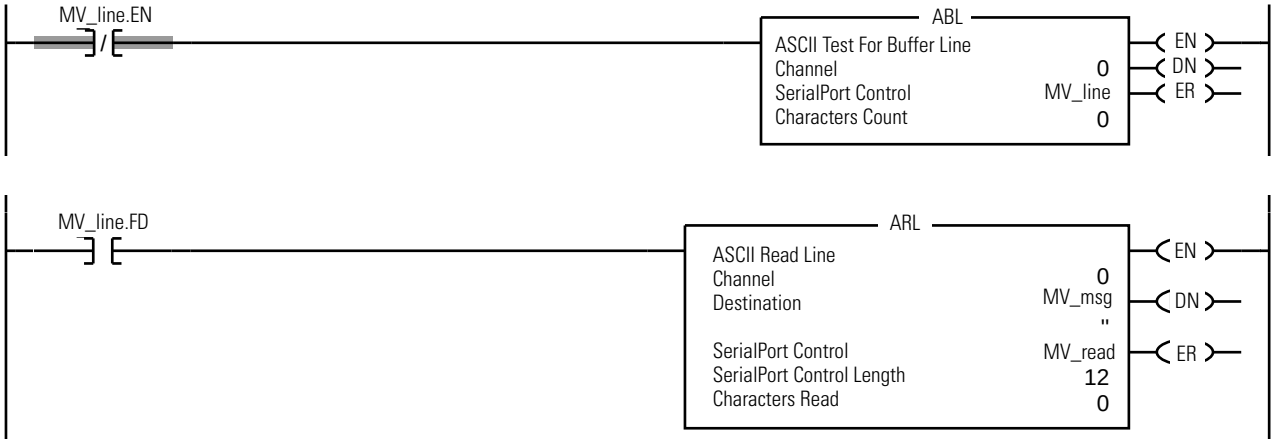
Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	Quando la condizione del ramo passa da azzerata a impostata, l'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzione	L'istruzione elimina i caratteri specificati dal buffer e li memorizza nella Destination. Il bit .EN viene impostato. I rimanenti bit di stato, tranne .UL, vengono azzerati. L'istruzione tenta di immettersi nella coda ASCII.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Verifica continuamente il buffer per un messaggio dal terminale MessageView. Poiché ciascun messaggio termina con un ritorno a capo ($r), il ritorno a capo è configurato come il carattere di terminazione nella finestra di dialogo Proprietà Controller della scheda Protocollo Utente. Quando l'ABL trova un ritorno a capo, imposta il bit FD.

Quando l'istruzione ABL trova il ritorno a capo (*MV_line.FD* è impostato), il controllore ha ricevuto un messaggio completo. L'istruzione ARL rimuove i caratteri dal buffer, fino al ritorno a capo incluso, e li posiziona nel membro DATA del tag *MV_msg*, che è una stringa.

Ladder



Testo strutturato

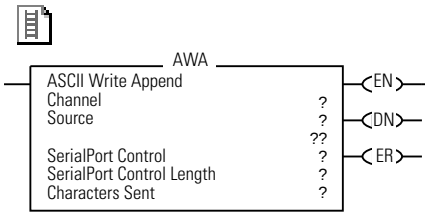
```
ABL(0,MV_line);


osri_1.InputBit := MVLine.FD;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
     mv_read.LEN := 12;
     ARL(0,MV_msg,MV_read);
END_IF;
```

ASCII Write Append (AWA)

L'istruzione AWA invia un numero specificato di caratteri del Tag Source ad un dispositivo seriale ed aggiunge uno o due caratteri predefiniti.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Channel	DINT	immediato tag	0	
Source	**stringa** SINT INT DINT	tag	tag che contiene i caratteri da inviare: • Per un tipo di dati stringa, inserire il nome del tag. • Per una matrice SINT, INT, o DINT, inserire il primo elemento della matrice.	• Se si vuole confrontare, convertire o manipolare i caratteri, usare un tipo di dati stringa. • I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Serial Port Control	SERIAL_PORT_CONTROL	tag	tag che controlla l'operazione	
Serial Port Control Length	DINT	immediato	numero di caratteri da inviare	• La Serial Port Control Length deve essere inferiore o uguale alla dimensione della Source. • Se si vuole impostare la Serial Port Control Length uguale al numero di caratteri nella Source, digitare 0.
Characters Sent	DINT	immediato	0	Durante l'esecuzione, visualizza il numero di caratteri che sono stati inviati.

Testo strutturato

```
AWA(Channel,Source,
    SerialPortControl);
```

Gli operandi sono gli stessi dell'istruzione AWA in ladder. Tuttavia, i valori Serial Port Control Length e Characters Sent vengono indicati accedendo ai membri .LEN e .POS della struttura SERIAL_PORT_CONTROL invece di inserire i valori nella lista di operandi.

Struttura SERIAL_PORT_CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione è abilitata.
.EU	BOOL	Il bit di coda indica che l'istruzione è entrata nella coda ASCII.
.DN	BOOL	Il bit di fine indica quando l'istruzione è stata eseguita, però è asincrono rispetto alla scansione logica.
.RN	BOOL	Il bit di esecuzione indica che l'istruzione è in corso di esecuzione.
.EM	BOOL	Il bit di vuoto indica quando l'istruzione è stata eseguita, però è sincrono rispetto alla scansione logica.
.ER	BOOL	Il bit di errore indica quando l'istruzione fallisce (genera un errore).
.FD	BOOL	Il bit di trovato non riguarda questa istruzione.
.LEN	DINT	La lunghezza indica il numero di caratteri da inviare.
.POS	DINT	La posizione visualizza il numero di caratteri che sono stati inviati.
.ERROR	DINT	L'errore contiene un valore esadecimale che identifica la causa di un errore.

Descrizione: L'istruzione AWA:

- invia il numero di caratteri specificato (Serial Port Control Length) del Tag Source al dispositivo collegato alla porta seriale del controllore
- aggiunge alla fine dei caratteri uno o due caratteri definiti nella finestra di dialogo Proprietà Controller della scheda Protocollo Utente

Per programmare un'istruzione AWA, seguire questa procedura:

1. Configurare la porta seriale del controllore:

   a. L'applicazione include anche delle istruzioni ARD o ARL?

Se	Allora
sì	Selezionare la modalità Utente
no	Selezionare o la modalità Utente o la modalità Sistema

   b. Definire i caratteri da aggiungere ai dati.

2. Questa è un'istruzione di transizione:
   - In ladder, cambia la condizione del ramo di ingresso da azzerata a impostata ogni volta che l'istruzione deve essere eseguita.
   - In testo strutturato, condiziona l' istruzione in modo che venga eseguita solo su una transizione.

3. Ogni volta che l'istruzione viene eseguita, inviare sempre lo stesso numero di caratteri?

Se	Allora
sì	Nella Serial Port Control Length, inserire il numero di caratteri da inviare.
no	Prima che l'istruzione venga eseguita, impostare il membro LEN del tag della Source al valore del membro LEN del tag Serial Port Control.

Indicatori di stato aritmetico: non influenzati

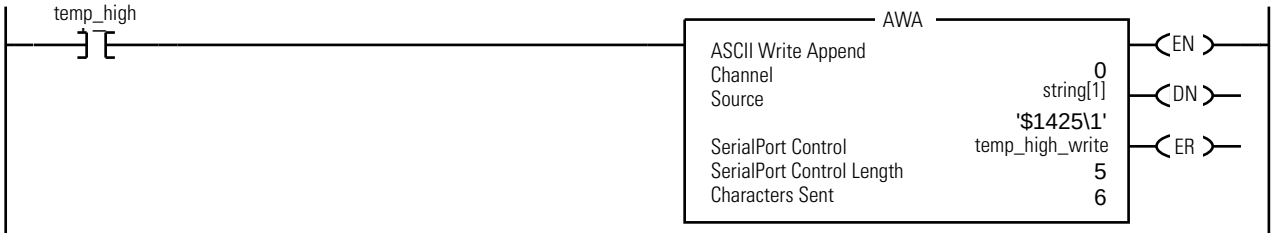
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	Quando la condizione del ramo passa da azzerata a impostata, l'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione invia un numero specificato di caratteri ed aggiunge uno o due caratteri predefiniti. Il bit .EN viene impostato. I rimanenti bit di stato, tranne .UL, vengono azzerati. L'istruzione tenta di immettersi nella coda ASCII.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio 1: Quando la temperatura supera il limite massimo superiore (*temp_high* è impostato), l'istruzione AWA invia un messaggio al terminale MessageView collegato alla porta seriale del controllore. Il messaggio contiene cinque caratteri dal membro DATA del tag *string[1]*, che è una stringa. (Il *$14* conta come un carattere. E' il codice esadecimale per il carattere Ctrl-T.) L'istruzione, inoltre, invia (aggiunge) i caratteri definiti nelle proprietà del controllore. In questo esempio, l'istruzione AWA invia un ritorno a capo ($0D) indicante la fine del messaggio.

Ladder

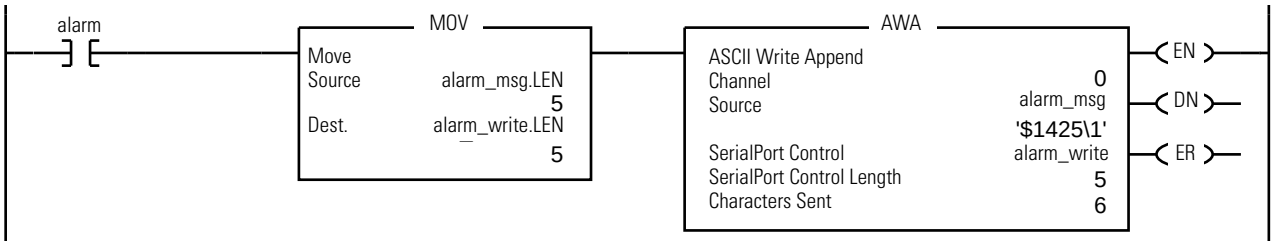


Testo strutturato

```
IF temp_high THEN

     temp_high_write.LEN := 5;

     AWA(0,string[1],temp_high_write);

     temp_high := 0;

END_IF;
```

Esempio 2: Quando *alarm* è attivo, l'istruzione AWA invia il numero specificato di caratteri in *alarm_msg* ed aggiunge uno o più caratteri di terminazione. Poiché il numero di caratteri in *alarm_msg* varia, il ramo sposta innanzitutto la lunghezza della stringa (*alarm_msg*.LEN) nella Serial Port Control Length dell'istruzione AWA (*alarm_write*.LEN). In *alarm_msg*, il *$14* conta come un carattere. E' il codice esadecimale per il carattere Ctrl-T.

Ladder



Testo strutturato

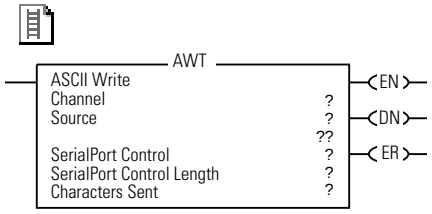
```
osri_1.InputBit := alarm;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
     alarm_write.LEN := alarm_msg.LEN;
     AWA(0,alarm_msg,alarm_write);
END_IF;
```

ASCII Write (AWT)

L'istruzione AWT invia un numero specificato di caratteri del Tag Source ad un dispositivo seriale.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Channel	DINT	immediato tag	0	
Source	**stringa** SINT INT DINT	tag	tag che contiene i caratteri da inviare: • Per un tipo di dati stringa, inserire il nome del tag. • Per una matrice SINT, INT, o DINT, inserire il primo elemento della matrice.	• Se si vuole confrontare, convertire o manipolare i caratteri, usare un tipo di dati stringa. • I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag che controlla l'operazione	
Serial Port Control Lenght	DINT	immediato	numero di caratteri da inviare	• La Serial Port Control Length deve essere inferiore o uguale alla dimensione della Source. • Se si vuole impostare la Serial Port Control Length uguale al numero di caratteri nella Source, digitare 0.
Characters Sent	DINT	immediato	0	Durante l'esecuzione, visualizza il numero di caratteri che sono stati inviati.

Testo strutturato

```
AWT(Channel,Source,
    SerialPortControl);
```

Gli operandi sono gli stessi dell'istruzione AWT in ladder. Tuttavia, i valori Serial Port Control Length e Characters Sent vengono indicati accedendo ai membri .LEN e .POS della struttura SERIAL_PORT_CONTROL invece di inserire i valori nella lista di operandi.

Struttura SERIAL_PORT_CONTROL

Mnemonico	Tipo di dati	Descrizione
.EN	BOOL	Il bit di abilitazione indica che l'istruzione è abilitata.
.EU	BOOL	Il bit di coda indica che l'istruzione è entrata nella coda ASCII.
.DN	BOOL	Il bit di fine indica quando l'istruzione è stata eseguita, però è asincrono rispetto alla scansione logica.
.RN	BOOL	Il bit di esecuzione indica che l'istruzione è in corso di esecuzione.
.EM	BOOL	Il bit di vuoto indica quando l'istruzione è stata eseguita, però è sincrono rispetto alla scansione logica.
.ER	BOOL	Il bit di errore indica quando l'istruzione fallisce (genera un errore).
.FD	BOOL	Il bit di trovato non riguarda questa istruzione.
.LEN	DINT	La lunghezza indica il numero di caratteri da inviare.
.POS	DINT	La posizione visualizza il numero di caratteri che sono stati inviati.
.ERROR	DINT	L'errore contiene un valore esadecimale che identifica la causa di un errore.

Descrizione: L'istruzione AWT invia il numero di caratteri specificato (Serial Port Control Length) del tag Source al dispositivo collegato alla porta seriale del controllore

Per programmare un'istruzione AWT, seguire questa procedura:

1. Configurare la porta seriale del controllore:

Se l'applicazione	Allora
usa istruzioni ARD o ARL	Selezionare la modalità Utente
non usa le istruzioni ARD o ARL	Selezionare o la modalità Utente o la modalità Sistema

2. Questa è un'istruzione di transizione:
   - In ladder, cambia la condizione del ramo di ingresso da azzerata a impostata ogni volta che l'istruzione deve essere eseguita.
   - In testo strutturato, condiziona l' istruzione in modo che venga eseguita solo su una transizione.

3. Ogni volta che l'istruzione viene eseguita, inviare sempre lo stesso numero di caratteri?

Se	Allora
sì	Nella Serial Port Control Length, inserire il numero di caratteri da inviare.
no	Prima che l'istruzione venga eseguita, spostare il membro LEN del tag della Source nel membro LEN del tag Serial Port Control.

Indicatori di stato aritmetico: non influenzati

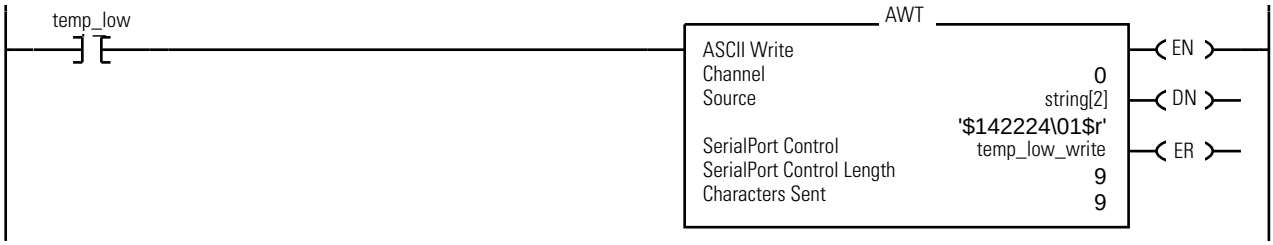
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	Quando la condizione del ramo passa da azzerata a impostata, l'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione invia un numero specificato di caratteri. Il bit .EN viene impostato. I rimanenti bit di stato, tranne .UL, vengono azzerati. L'istruzione tenta di immettersi nella coda ASCII.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio 1: Quando la temperatura raggiunge il limite inferiore (*temp_low* è impostato), l'istruzione AWT invia un messaggio al terminale MessageView connesso alla porta seriale del controllore. Il messaggio contiene nove caratteri provenienti dal membro DATA del tag *string[2]*, che è una stringa. (Il *$14* conta come un carattere. E' il codice esadecimale per il carattere Ctrl-T.) L'ultimo carattere è un ritorno a capo ($r), che indica la fine del messaggio.

Ladder

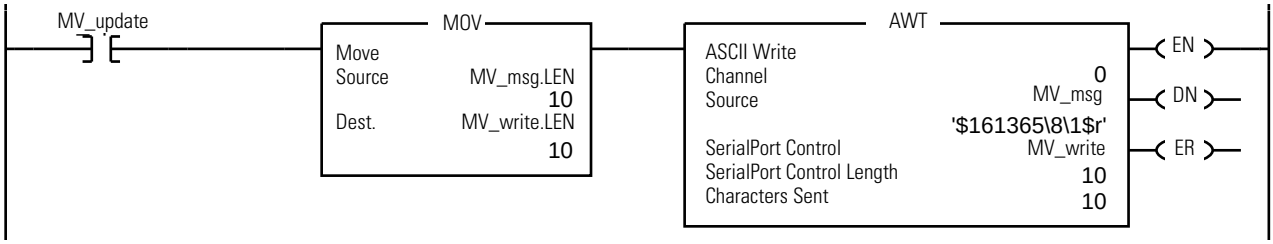


Testo strutturato

```
osri_1.InputBit := temp_low;
OSRI(osri_1);
IF (osri_1.OutputBit) THEN
     temp_low_write.LEN := 9;
     AWT(0,string[2],temp_low_write);
END_IF;
```

Esempio 2: Quando *MV_update* è impostato, l'istruzione AWT invia i caratteri in *MV_msg*. Poiché il numero di caratteri in *MV_msg* varia, il ramo sposta innanzitutto la lunghezza della stringa (*MV_msg*.LEN) nella Serial Port Control Length dell'istruzione AWT (*MV_write*.LEN). In *MV_msg*, il *$16* conta come un carattere. E' il codice esadecimale per il carattere Ctrl-V.

Ladder



Testo strutturato

```
osri_1.InputBit := MV_update;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
     MV_write.LEN := Mv_msg.LEN;
     AWT(0,MV_msg,MV_write);
END_IF;
```

Note:

Istruzioni di stringhe ASCII
(CONCAT, DELETE, FIND, INSERT, MID)

Introduzione

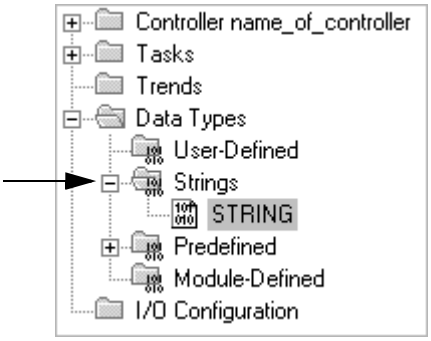
Utilizzare le istruzioni di stringhe ASCII per modificare e creare stringhe di caratteri ASCII .

Se si desidera	Per esempio	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
aggiungere caratteri alla fine di una stringa	aggiungere caratteri di terminazione o delimitatori ad una stringa	CONCAT	ladder testo strutturato	593
eliminare caratteri da una stringa	rimuovere caratteri di testa o di controllo da una stringa	DELETE	ladder testo strutturato	595
determinare il carattere di partenza di una sottostringa	localizza un gruppo di caratteri all'interno di una stringa	FIND	ladder testo strutturato	597
inserire caratteri in una stringa	crea una stringa che utilizza variabili	INSERT	ladder testo strutturato	599
estrarre caratteri da una stringa	estrarre informazioni da un codice a barre	MID	ladder testo strutturato	601

Per confrontare o convertire caratteri ASCII si possono anche utilizzare le seguenti istruzioni:

Se si desidera	Utilizzare questa istruzione	Vedere pagina
confrontare una stringa ad un'altra	CMP	204
vedere se i caratteri sono uguali a caratteri specifici	EQU	209
vedere se i caratteri sono diversi da caratteri specifici	NEQ	240
vedere se i caratteri sono uguali o superiori a caratteri specifici	GEQ	213
vedere se i caratteri sono superiori a caratteri specifici	GRT	217
vedere se i caratteri sono uguali o inferiori a caratteri specifici	LEQ	221
vedere se i caratteri sono inferiori a caratteri specifici	LES	225
riposizionare i byte di un tag INT, DINT, o REAL	SWPB	298
trovare una stringa in una matrice di stringhe	FSC	342
convertire caratteri in un valore SINT, INT, DINT, o REAL	STOD	606
convertire caratteri in un valore REAL	STOR	608
convertire un valore SINT, INT, DINT o REAL in una stringa di caratteri ASCII	DTOS	611
convertire un valore REAL in una stringa di caratteri ASCII	RTOS	613

Tipi di dati stringa



Si memorizzano caratteri ASCII in tag che utilizzano un tipo di dati stringa.

- Si può utilizzare il tipo di dati STRINGA di default. Esso memorizza fino a 82 caratteri.
- Si può creare un nuovo tipo di dati stringa che memorizza più o meno caratteri.

Per creare un nuovo tipo di dati stringa, vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.

Ciascun tipo di dati stringa contiene i seguenti membri:

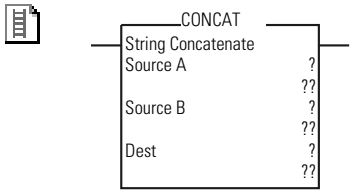
Nome	Tipo di dati	Descrizione	Note
LEN	DINT	numero di caratteri nella stringa	LEN si aggiorna automaticamente al nuovo conteggio dei caratteri ogni volta che: • si usa la finestra di dialogo Browser Stringa per inserire caratteri • si usano istruzioni che leggono, convertono o manipolano una stringa LEN mostra la lunghezza della stringa attuale. Il membro DATA può contenere caratteri vecchi addizionali che non sono inclusi nel conteggio LEN.
DATA	Matrice SINT	caratteri ASCII della stringa	• Per accedere ai caratteri della stringa, indirizzare il nome del tag. Per esempio, per accedere ai caratteri della *string_1*, inserire *string_1*. • Ciascun elemento della matrice DATA contiene un carattere. • Si possono creare nuovi tipi di dati stringa che memorizzano più o meno caratteri.

String Concatenate (CONCAT)

L'istruzione CONCAT aggiunge caratteri ASCII alla fine di una stringa.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Source A	stringa	tag	tag che contiene i caratteri iniziali	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Source B	stringa	tag	tag che contiene i caratteri finali	
Destination	stringa	tag	tag dove memorizzare il risultato	

Testo strutturato

```
CONCAT(SourceA,SourceB,
    Dest);
```

Gli operandi sono gli stessi dell'istruzione CONCAT in ladder.

Descrizione:

L'istruzione CONCAT combina i caratteri di Source A con quelli di Source B e colloca i risultati nella Destination.

- I caratteri da Source A vengono prima, seguiti dai caratteri di Source B.
- A meno che Source A e Destination non siano lo stesso tag, Source A rimane invariata.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione concatena le stringhe.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Per lanciare un messaggio in un terminale MessageView, il controllore deve inviare una stringa ASCII che contiene un numero messaggio ed un numero nodo. *String_1* contiene il numero messaggio. Quando *add_node* è impostato, l'istruzione CONCAT aggiunge i caratteri in *node_num_ascii* (numero nodo) alla fine dei caratteri in *string_1* e quindi memorizza il risultato in *msg*.

Ladder

Testo strutturato

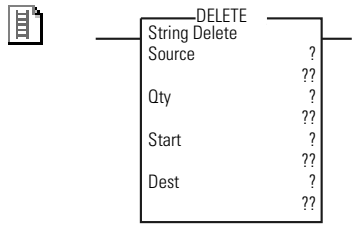
```
IF add_node THEN

     CONCAT(string_1,node_num_ascii,msg);

     add_node := 0;

END_IF;
```

String Delete (DELETE)

L'istruzione DELETE rimuovere caratteri ASCII da una stringa.

Operandi:

Ladder



Operando	Tipo	formato	Digitare	Note
Source	stringa	tag	tag che contiene la stringa dalla quale si vogliono eliminare caratteri	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Quantity	SINT INT **DINT**	immediato tag	numero di caratteri da eliminare	Start più la quantità deve essere minore o uguale alla dimensione di DATA della Source.
Start	SINT INT **DINT**	immediato tag	posizione del primo carattere da eliminare	Inserire un numero tra 1 e la dimensione di DATA della Source.
Destination	stringa	tag	tag dove memorizzare il risultato	

Testo strutturato

```
DELETE(Source,Qty,Start,
     Dest);
```

Gli operandi sono gli stessi dell'istruzione DELETE in ladder.

Descrizione: L'istruzione DELETE elimina (rimuove) un gruppo di caratteri dalla Source e posiziona i caratteri rimanenti nella Destination.

- La posizione Start e Quantity definiscono i caratteri da rimuovere.
- A meno che Source e Destination non siano lo stesso tag, la Source rimane invariata.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

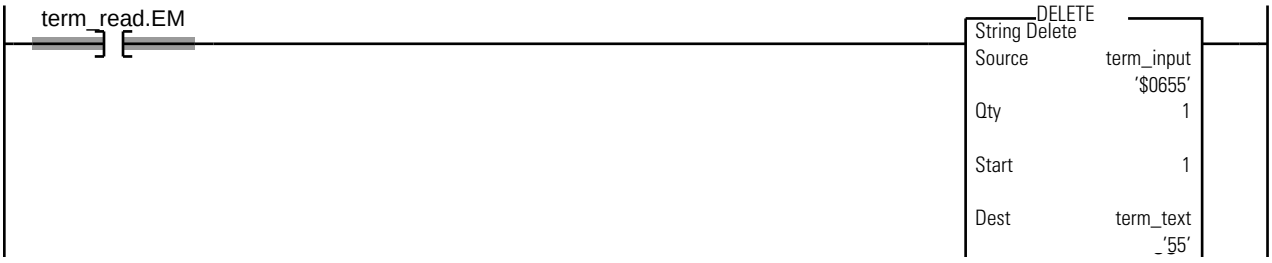
Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.
4	56	Il valore Start o Quantity non è valido.	1. Verificare che il valore Start sia compreso tra 1 e la dimensione di DATA della Source. 2. Verificare che il valore di Start più la Quantity sia minore o uguale alla dimensione di DATA della Source.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione elimina i caratteri specificati.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Informazioni ASCII da un terminale contengono un carattere di intestazione. Dopo che il controllore ha letto i dati (*term_read.EM* è impostato) l'istruzione DELETE rimuove il carattere di intestazione.

Ladder



Testo strutturato

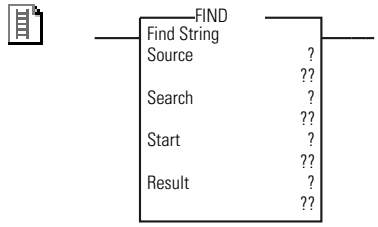
```
IF term_read.EM THEN
     DELETE(term_input,1,1,term_text);
     term_read.EM := 0;
END_IF;
```

Find String (FIND)

L'istruzione FIND localizza la posizione di partenza di una stringa specificata all'interno di un'altra stringa

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Source	stringa	tag	stringa da ricercare in	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Search	stringa	tag	stringa da trovare	
Start	SINT INT **DINT**	immediato tag	posizione in Source per iniziare la ricerca	Inserire un numero tra 1 e la dimensione di DATA della Source.
Result	SINT INT **DINT**	tag	tag che memorizza la posizione di partenza della stringa da trovare	

Testo strutturato

```
FIND(Source,Search,Start,
     Result);
```

Gli operandi sono gli stessi dell'istruzione FIND di ladder sopra descritta.

Descrizione:

L'istruzione FIND ricerca la stringa Search nella stringa Source. Se l'istruzione trova la stringa Search, in corrispondenza di Result viene indicata la posizione iniziale della stringa da ricercare all'interno della stringa sorgente.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.
4	56	Il valore Start non è valido.	Verificare che il valore Start sia compreso tra 1 e la dimensione di DATA della Source.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione cerca i caratteri specificati.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Un messaggio dal terminale MessageView contiene diverse parti di informazione. Il carattere backslash [\ ] separa ciascuna parte di informazione. Per localizzare una parte di informazione, l'istruzione FIND ricerca il carattere backslash e registra la sua posizione in *find_pos*.

Ladder

Testo strutturato

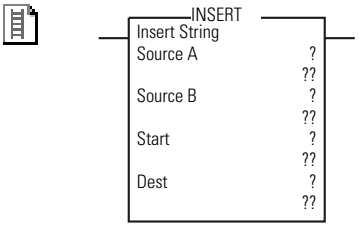
```
IF MV_read.EM THEN

     FIND(MV_msg,find,1,find_pos);

     MV_read.EM := 0;

END_IF;
```

Insert String (INSERT)

L'istruzione INSERT aggiunge caratteri ASCII in una determinata posizione all'interno di una stringa.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Source A	stringa	tag	stringa a cui aggiungere caratteri	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Source B	stringa	tag	stringa che contiene i caratteri da aggiungere	
Start	SINT INT **DINT**	immediato tag	posizione in Source A per aggiungere i caratteri	Inserire un numero tra 1 e la dimensione di DATA della Source.
Result	stringa	tag	stringa per memorizzare il risultato	

Testo strutturato

```
INSERT(SourceA,SourceB,
    Start,Dest);
```

Gli operandi sono gli stessi dell'istruzione INSERT in ladder.

Descrizione:

L'istruzione INSERT somma i caratteri di Source A con quelli di Source B in una posizione specificata all'interno di Source A e colloca i risultati nella Destination.

- Start definisce dove in Source A viene aggiunto Source B.
- A meno che Source A e la Destination non siano lo stesso tag, Source A rimane invariata.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.
4	56	Il valore Start non è valido.	Verificare che il valore Start sia compreso tra 1 e la dimensione di DATA della Source.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione inserisce i caratteri specificati.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Quando *temp_high* è impostato, l'istruzione INSERT aggiunge i caratteri in *string_2* alla posizione 2 all'interno di *string_1* e posiziona il risultato in *string_3*:

Ladder

Testo strutturato

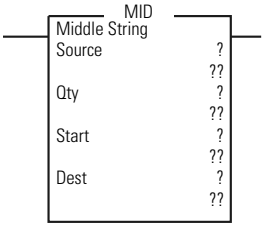
```
IF temp_high THEN

     INSERT(string_1,string_2,2,string_3);

     temp_high := 0;

END_IF;
```

Middle String (MID)

L'istruzione MID copia uno specificato numero di caratteri ASCII da una stringa e li memorizza in un'altra stringa.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Source	stringa	tag	stringa da cui copiare caratteri	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Quantity	SINT INT **DINT**	immediato tag	numero di caratteri da copiare	Start più Quantity deve essere minore o uguale alla dimensione di DATA della Source.
Start	SINT INT **DINT**	immediato tag	posizione del primo carattere da copiare	Inserire un numero tra 1 e la dimensione di DATA della Source.
Destination	stringa	tag	stringa in cui copiare caratteri	

Testo strutturato

```
MID(Source,Qty,Start,
    Dest);
```

Gli operandi sono gli stessi dell'istruzione MID in ladder.

Descrizione: L'istruzione MID copia un gruppo di caratteri dalla Source e posiziona il risultato nella Destination.

- La posizione di partenza e la quantità definiscono i caratteri da copiare.
- A meno che Source e Destination non siano lo stesso tag, la Source rimane invariata.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

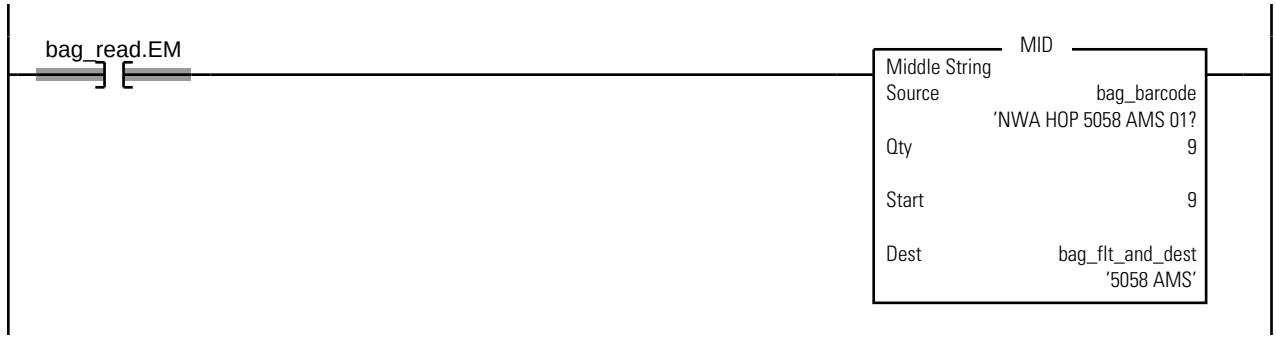
Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.
4	56	Il valore Start o Quantity non è valido.	1. Verificare che il valore Start sia compreso tra 1 e la dimensione di DATA della Source. 2. Verificare che il valore di Start più Quantity sia minore o uguale alla dimensione di DATA della Source.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione copia i caratteri specificati da una stringa e li memorizza in un'altra stringa.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Nel trasportatore dei bagagli di un aeroporto, su ciascun bagaglio viene applicato un codice a barre. I caratteri 9–17 del codice a barre sono il numero del volo e l'aeroporto di destinazione del bagaglio. Dopo che il codice a barre è stato letto, (*bag_read.EM* è impostato) l'istruzione MID copia il numero del volo e l'aeroporto di destinazione nella stringa *bag_flt_and_dest*.

Ladder



Testo strutturato

```
IF bag_read.EM THEN
     MID(bar_barcode,9,9,bag_flt_and_dest);
     bag_read.EM := 0;
END_IF;
```

Istruzioni di conversione ASCII
(STOD, STOR, DTOS, RTOS, UPPER, LOWER)

Introduzione

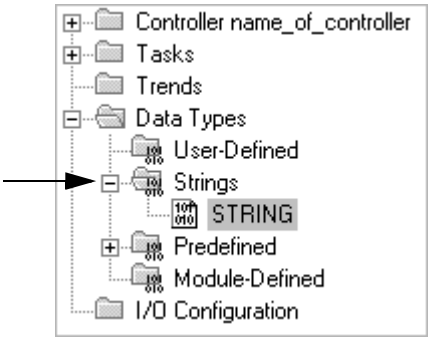
Le istruzioni di conversione ASCII servono per modificare il formato dei dati.

Se si desidera	Per esempio	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
convertire la rappresentazione ASCII di un valore intero in valore SINT, INT, DINT, o REAL	convertire un valore da una bilancia o altro dispositivo ASCII in un intero, così da poterlo utilizzare nella logica	STOD	ladder testo strutturato	606
convertire la rappresentazione ASCII di un valore a virgola mobile in un valore REAL	convertire un valore da una bilancia o altro dispositivo ASCII in un valore REAL così da poterlo utilizzare nella logica	STOR	ladder testo strutturato	608
convertire un valore SINT, INT, DINT o REAL in una stringa di caratteri ASCII	convertire una variabile in una stringa ASCII così da poterla inviare ad un terminale MessageView	DTOS	ladder testo strutturato	611
convertire un valore REAL in una stringa di caratteri ASCII	convertire una variabile in una stringa ASCII così da poterla inviare ad un terminale MessageView	RTOS	ladder testo strutturato	613
convertire le lettere di una stringa di caratteri ASCII in maiuscolo.	convertire un dato inserito dall'operatore in maiuscolo per poterlo ricercare in una matrice	UPPER	ladder testo strutturato	615
convertire le lettere di una stringa di caratteri ASCII in minuscolo.	convertire un dato inserito dall'operatore in minuscolo per poterlo ricercare in una matrice	LOWER	ladder testo strutturato	617

Per confrontare o manipolare caratteri ASCII si possono anche utilizzare le seguenti istruzioni:

Se si desidera	Utilizzare questa istruzione	Vedere pagina
aggiungere caratteri alla fine di una stringa	CONCAT	593
eliminare caratteri da una stringa	DELETE	595
determinare il carattere di partenza di una sottostringa	FIND	597
inserire caratteri in una stringa	INSERT	599
estrarre caratteri da una stringa	MID	601
riposizionare i byte di un tag INT, DINT, o REAL	SWPB	298
confrontare una stringa ad un'altra	CMP	204
vedere se i caratteri sono uguali a caratteri specifici	EQU	209
vedere se i caratteri sono diversi da caratteri specifici	NEQ	240
vedere se i caratteri sono uguali o superiori a caratteri specifici	GEQ	213
vedere se i caratteri sono superiori a caratteri specifici	GRT	217
vedere se i caratteri sono uguali o inferiori a caratteri specifici	LEQ	221
vedere se i caratteri sono inferiori a caratteri specifici	LES	225
trovare una stringa in una matrice di stringhe	FSC	342

Tipi di dati stringa



Si memorizzano caratteri ASCII in tag che utilizzano un tipo di dati stringa.

- Si può utilizzare il tipo di dati STRINGA di default. Esso memorizza fino a 82 caratteri.
- Si può creare un nuovo tipo di dati stringa che memorizza più o meno caratteri.

Per creare un nuovo tipo di dati stringa, vedere *Logix5000 Controllori Logix5000 – Procedure comuni*, pubblicazione 1756-PM001.

Ciascun tipo di dati stringa contiene i seguenti membri:

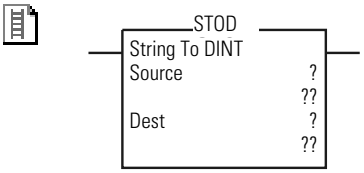
Nome	Tipo di dati	Descrizione	Note
LEN	DINT	numero di caratteri nella stringa	LEN si aggiorna automaticamente al nuovo conteggio dei caratteri ogni volta che: • si usa la finestra di dialogo Browser Stringa per inserire caratteri • si usano istruzioni che leggono, convertono o manipolano una stringa LEN mostra la lunghezza della stringa attuale. Il membro DATA può contenere caratteri vecchi addizionali che non sono inclusi nel conteggio LEN.
DATA	Matrice SINT	caratteri ASCII della stringa	• Per accedere ai caratteri della stringa, indirizzare il nome del tag. Per esempio, per accedere ai caratteri della *string_1*, inserire *string_1*. • Ciascun elemento della matrice DATA contiene un carattere. • Si possono creare nuovi tipi di dati stringa che memorizzano più o meno caratteri.

String To DINT (STOD)

L'istruzione STOD converte la rappresentazione ASCII di un intero in un valore intero o REAL.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Source	stringa	tag	tag che contiene il valore in ASCII	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Destination	SINT INT **DINT** REAL	tag	tag in cui memorizzare il valore intero	Se il valore Source è un numero in virgola mobile, l'istruzione converte soltanto la parte non frazionaria del numero (a prescindere dal tipo di dati di destinazione).

Testo strutturato

```
STOD(Source,Dest);
```

Gli operandi sono gli stessi dell'istruzione STOD in ladder.

Descrizione: La STOD converte la Source in un numero intero ed inserisce il risultato in Destination.

- L'istruzione converte numeri positivi e negativi.
- Se la stringa Source contiene caratteri non numerici, STOD converte la prima serie di numeri contigui:
  - L'istruzione salta ogni carattere di controllo o non numerico iniziale (eccetto il segno meno davanti ad un numero).
  - Se la stringa contiene gruppi multipli di numeri separati da delimitatori (per esempio, /), l'istruzione converte soltanto il primo gruppo di numeri.

Indicatori di stato aritmetico: gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore

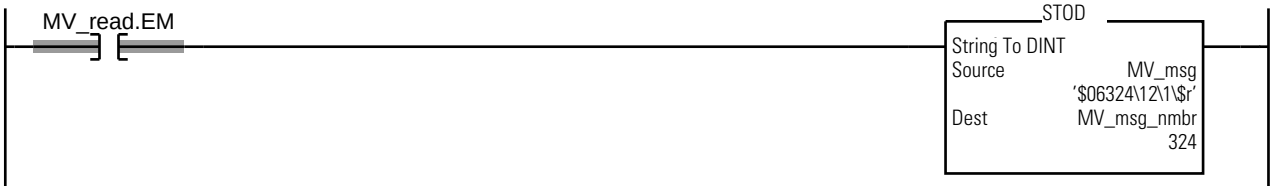
Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.
4	53	Il numero in output supera i limiti del tipo di dato destinazione.	Uno o l'altro: • Ridurre la dimensione del valore ASCII. • Utilizzare un tipo di dati maggiore per la destinazione.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	SC viene impostato. La destinazione è cancellata. L'istruzione converte la Source. Se il risultato è zero, allora viene impostato S:Z	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Quando *MV_read.EM* è impostato, l'istruzione STOD converte la prima serie di caratteri numerici in *MV_msg* in un valore intero. L'istruzione salta il carattere di controllo iniziale ($06) e si interrompe al delimitatore (\ ).

Ladder



Testo strutturato

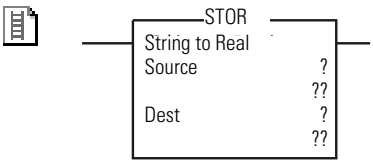
```
IF MV_read.EM THEN
     STOD(MV_msg,MV_msg_nmbr);
     MV_read.EM := 0;
END_IF;
```

String To REAL (STOR)

L'istruzione STOR converte la rappresentazione ASCII di un valore in virgola mobile in un valore REAL.

Operandi:

Operandi ladder



Operando	Tipo	Formato	Digitare	Note
Source	stringa	tag	tag che contiene il valore in ASCII	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea
Destination	REAL	tag	tag in cui memorizzare il valore REAL	

Testo strutturato

```
STOR(Source,Dest);
```

Gli operandi sono gli stessi dell'istruzione STOR in ladder.

Descrizione:

STOR converte il valore Source in valore REAL ed inserisce il risultato in Destination.

- L'istruzione converte numeri positivi e negativi.
- Se la stringa Source contiene caratteri non numerici, STOR converte la prima serie di numeri contigui, incluso il punto decimale [.]:
  - L'istruzione salta ogni carattere di controllo o non numerico iniziale (eccetto il segno meno davanti ad un numero).
  - Se la stringa contiene gruppi multipli di numeri separati da delimitatori (ad esempio, /), l'istruzione converte soltanto il primo gruppo di numeri.

Indicatori di stato aritmetico:

gli indicatori di stato aritmetici sono influenzati.

Condizioni di errore:

Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.
4	53	Il numero in output supera i limiti del tipo di dato destinazione.	Uno o l'altro: • Ridurre la dimensione del valore ASCII. • Utilizzare un tipo di dati maggiore per la destinazione.

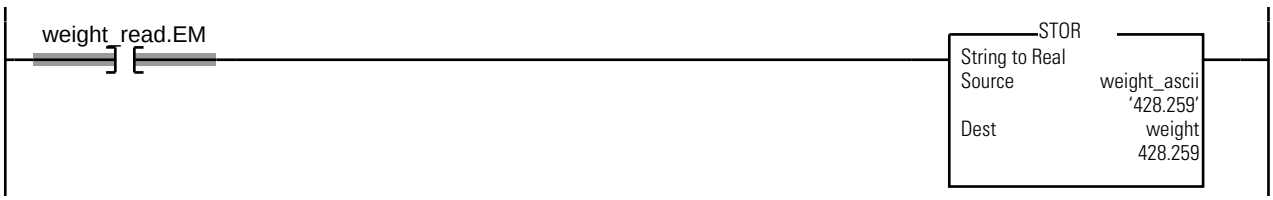
Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	S:C è impostato. La destinazione è cancellata. L'istruzione converte la Source. Se il risultato è zero, allora viene impostato S:Z	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Dopo aver letto il peso da una bilancia (*weight_read.EM* è impostato) l'istruzione STOR converte i caratteri numerici in *weight_ascii* in un valore REAL.

Si può riscontrare una leggera differenza tra le parti frazionarie di Source e Destination.

Ladder



Testo strutturato

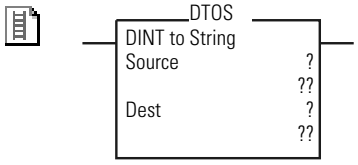
```
IF weight_read.EM THEN
     STOR(weight_ascii,weight);
     weight_read.EM := 0;
END_IF;
```

DINT to String (DTOS)

L'istruzione DTOS produce la rappresentazione ASCII di un valore.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Source	SINT INT **DINT** REAL	tag	tag che contiene il valore	Se la Source è un REAL, l'istruzione lo converte in un valore DINT. Consultare Da REAL a intero a pagina 632.
Destination	stringa	tag	tag in cui memorizzare il valore ASCII	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea

Testo strutturato

```
DTOS(Source,Dest);
```

Gli operandi sono gli stessi dell'istruzione DTOS in ladder.

Descrizione:

L'istruzione DTOS converte la Source in una stringa di caratteri ASCII e posiziona il risultato nella Destination.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

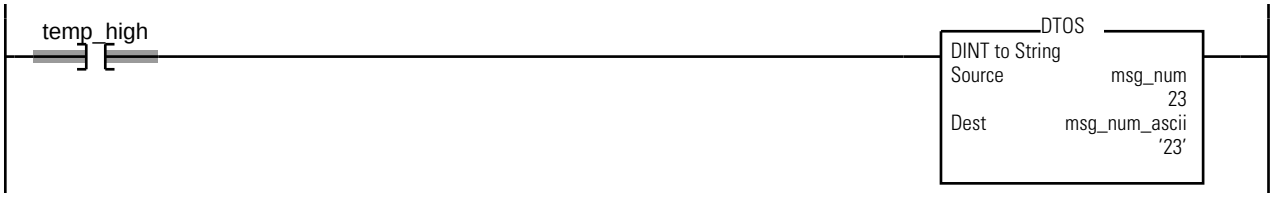
Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.
4	52	La stringa output è più grande della destinazione.	Creare un nuovo tipo di dati stringa abbastanza grande per la stringa output. Utilizzare il nuovo tipo di dati stringa come tipo di dati per la destinazione.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione converte Source.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Quando *temp_high* è impostato, l'istruzione DTOS converte il valore in *msg_num* in una stringa di caratteri ASCII e posiziona il risultato in *msg_num_ascii*. I rami successivi inseriscono o concatenano *msg_num_ascii* con altre stringhe per produrre un messaggio completo per il display del terminale.

Ladder



Testo strutturato

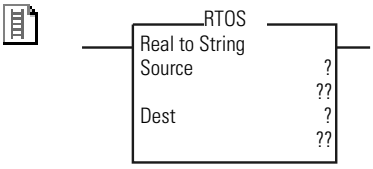
```
IF temp_high THEN

     DTOS(msg_num,msg_num_ascii);

     temp_high := 0;

END_IF;
```

REAL to String (RTOS)

L'istruzione RTOS produce una rappresentazione ASCII di un valore REAL.

Operandi:

Ladder



Operando	Tipo	Formato	Digitare	Note
Source	REAL	tag	tag che contiene il valore REAL	
Destination	stringa	tag	tag in cui memorizzare il valore ASCII	I tipi di dati stringa sono: • tipo di dati STRINGA di default • qualunque nuovo tipo di dati stringa che si crea

Testo strutturato

```
RTOS(Source,Dest);
```

Gli operandi sono gli stessi dell'istruzione RTOS in ladder.

Descrizione: L'istruzione RTOS converte la Source in una stringa di caratteri ASCII e posiziona il risultato nella Destination.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

Tipo	Codice	Causa	Metodo di ripristino
4	51	Il valore LEN del tag stringa è maggiore della dimensione di DATA del tag stringa.	1. Verificare che nessuna istruzione sia scritta nel membro LEN del tag tipo di stringa. 2. Nel valore LEN, inserire il numero di caratteri contenuto dalla stringa.
4	52	La stringa output è più grande della destinazione.	Creare un nuovo tipo di dati stringa abbastanza grande per la stringa output. Utilizzare il nuovo tipo di dati stringa come tipo di dati per la destinazione.

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione converte la Source.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Quando *send_data* è impostato, l'istruzione RTOS converte il valore in *data_1* in una stringa di caratteri ASCII e posiziona il risultato in *data_1_ascii*. I rami successivi inseriscono o concatenano *data_1_ascii* con altre stringhe per produrre un messaggio completo per un display terminale.

Si può riscontrare una leggera differenza tra le parti frazionarie di Source e Destination.

Ladder

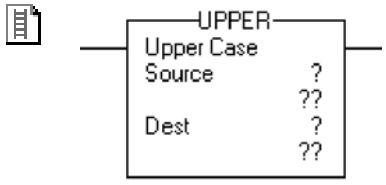
Testo strutturato

```
IF send_data THEN

     RTOS(data_1,data_1_ascii);

     send_data := 0;

END_IF;
```

Upper Case (UPPER)

L'istruzione UPPER converte i caratteri alfabetici di una stringa in caratteri maiuscoli.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source	stringa	tag	tag che contiene i caratteri da convertire in maiuscolo
Destination	stringa	tag	tag in cui memorizzare i caratteri in maiuscolo

```
UPPER(Source,Dest);
```

Testo strutturato

Gli operandi sono gli stessi dell'istruzione UPPER in ladder.

Descrizione: L'istruzione UPPER converte tutte le lettere della Source in maiuscolo e posiziona il risultato nella Destination.

- I caratteri ASCII sono sensibili alle maiuscole. "A" maiuscola ($41) non è uguale ad "a" minuscola ($61).
- Se si immettono direttamente i caratteri ASCII, convertire tutti i caratteri in maiuscolo o minuscolo prima di confrontarli.

Gli eventuali caratteri della stringa Source che non sono lettere rimarranno invariati.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione converte la Source in maiuscolo.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Per trovare informazioni in merito ad un articolo specifico, l'operatore immette il numero di catalogo dell'articolo in un terminale ASCII. Dopo che il controllore ha letto i dati forniti dal terminale (*terminal_read.EM* è impostato), l'istruzione UPPER converte tutti i caratteri di *catalog_number* in caratteri maiuscoli e memorizza il risultato in *catalog_number_upper_case*. Quindi un ramo successivo effettua una ricerca all'interno di una matrice per individuare eventuali caratteri corrispondenti a quelli di *catalog_number_upper_case*.

Ladder

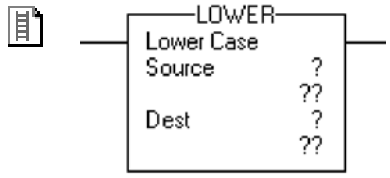
Testo strutturato

```
IF terminal_read.EM THEN

     UPPER(catalog_number,catalog_number_upper_case);

     terminal_read.EM := 0;

END_IF;
```

Lower Case (LOWER)

L'istruzione LOWER converte i caratteri alfabetici di una stringa in caratteri minuscoli.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Source	stringa	tag	tag che contiene i caratteri da convertire in minuscolo
Destination	stringa	tag	tag in cui memorizzare i caratteri in minuscolo

```
LOWER(Source,Dest);
```

Testo strutturato

Gli operandi sono gli stessi dell'istruzione LOWER in ladder.

Descrizione:

L'istruzione LOWER converte tutte le lettere della Source in minuscolo e posiziona il risultato nella Destination.

- I caratteri ASCII sono sensibili alle maiuscole. "A" maiuscola ($41) non è uguale ad "a" minuscola ($61).
- Se si immettono direttamente i caratteri ASCII, convertire tutti i caratteri in maiuscolo o minuscolo prima di confrontarli.

Gli eventuali caratteri della stringa Source che non sono lettere rimarranno invariati.

Indicatori di stato aritmetico:

non influenzati

Condizioni di errore:

nessuna

Esecuzione:

Condizione	Azione ladder	Azione testo strutturato
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.	na
se la condizione del ramo di ingresso è vera	L'istruzione viene eseguita. La condizione del ramo di uscita viene impostata su vero.	na
EnableIn viene impostato	na	EnableIn viene sempre impostato. L'istruzione viene eseguita.
esecuzione istruzioni	L'istruzione converte la Source in minuscolo.	
post-scansione	La condizione del ramo di uscita viene impostata su falso.	Nessuna.

Esempio: Per trovare informazioni in merito ad un articolo specifico, l'operatore immette il numero di catalogo dell'articolo in un terminale ASCII. Dopo che il controllore ha letto i dati forniti dal terminale (*terminal_read.EM* è impostato), l'istruzione LOWER converte tutti i caratteri di *item_number* in caratteri minuscoli e memorizza il risultato in *item_number_lower_case*. Quindi un ramo successivo effettua una ricerca all'interno di una matrice per individuare eventuali caratteri corrispondenti a quelli di *item_number_lower_case*.

Ladder

Testo strutturato

```
IF terminal_read.EM THEN

     LOWER(item_number,item_number_lower_case);

     terminal_read.EM := 0;

END_IF;
```

Istruzioni di debug
(BPT, TPT)

Introduzione

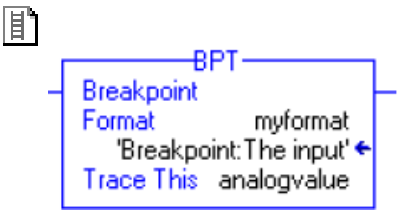
Usare le istruzioni di debug per monitorare lo stato della logica quando è in condizioni determinate dell'utente. Tali istruzioni sono compatibili solo con il software RSLogix Emulate 5000, con cui è possibile emulare un controllore Logix 5000 sul personal computer.

Se si desidera	Utilizzare questa istruzione	Disponibile in questi linguaggi	Vedere pagina
arrestare l'emulazione del programma quando un ramo è vero	BPT	ladder	619
registrare i dati selezionati quando un ramo è vero	TPT	ladder	623

Breakpoints (BPT)

I breakpoints fermano l'emulazione del programma quando un ramo è vero.

Operandi:



Ladder

Operando	Tipo	Formato	Descrizione
Format	stringa	tag	Una stringa che imposta la formattazione per il testo che compare nella finestra di tracciamento per il breakpoint.
Trace This	BOOL, SINT, INT, DINT, REAL	tag	Il tag che ha un valore che si desidera visualizzare nella finestra di tracciamento.

Descrizione:

I breakpoint sono programmati con l'istruzione di uscita Breakpoint (BPT). Quando gli ingressi di un ramo che contiene un'istruzione BTP sono veri, l'istruzione BTP interrompe l'esecuzione del programma. Il software visualizza una finestra che indica che il punto di arresto si è attivato e i valori che lo hanno attivato.

Quando viene attivato un breakpoint, l'emulatore visualizza una

finestra che informa che si è verificato un breakpoint. La barra del titolo della finestra mostra lo slot che contiene l'emulatore che ha incontrato il breakpoint.

Facendo clic su **OK**, l'emulatore riprende l'esecuzione del programma. Se le condizioni che hanno attivato il breakpoint persistono, il breakpoint si verifica nuovamente.

Inoltre, l'emulatore apre una finestra di tracciamento per il breakpoint, che visualizza le informazioni sul breakpoint e i valori.

ATTENZIONE

Quando viene attivato un breakpoint, non è possibile modificare il programma finché non si consente che l'esecuzione continui. È possibile passare online con l'emulatore per osservare lo stato del progetto, ma non sarà possibile modificarlo. Se si tenta di accettare una modificare del ramo mentre un breakpoint è attivato, compare una finestra di dialogo che comunica che il controllore non è nella modalità corretta.

Stringa Format

Con la stringa Format nelle istruzioni punto di tracciamento e breakpoint, è possibile controllare in che modo i tag tracciati appaiono nelle tracce o nelle finestre dei breakpoint. Il formato della stringa è mostrato di seguito:

intestazione:(testo)%(tipo)

dove *intestazione* è una stringa di testo che identifica il punto di tracciamento o il breakpoint, *testo* è una stringa che descrive il tag (o qualsiasi altro testo selezionato) e % (*tipo*) indica il formato del tag. Occorre un indicatore di tipo per ogni tag tracciato con l'istruzione punto di tracciamento o breakpoint.

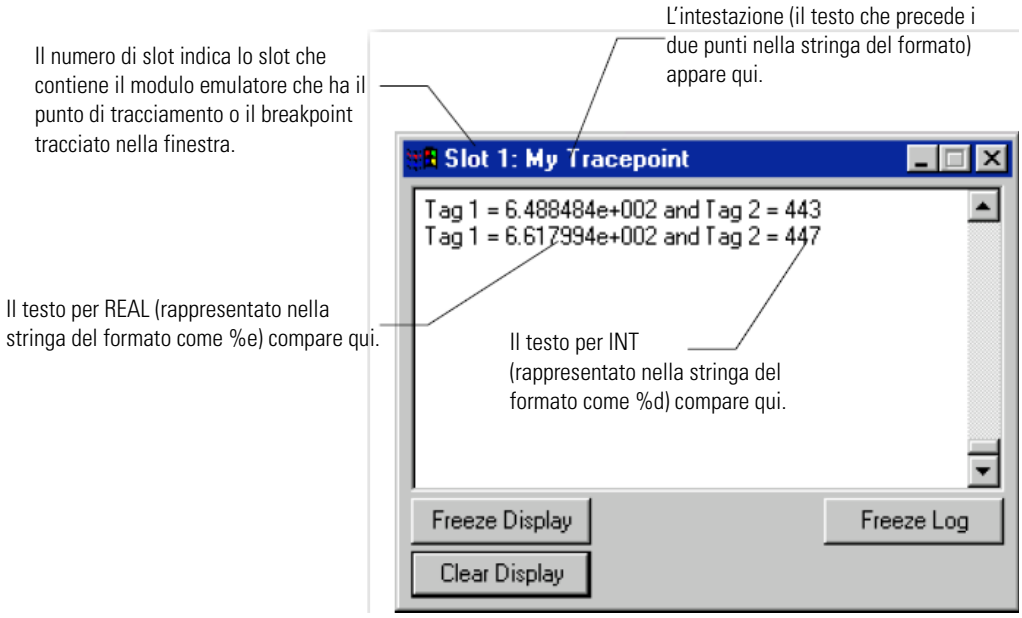
Ad esempio, è possibile formattare una stringa di un punto di tracciamento nel modo seguente:

My tracepoint:Tag 1 = %e e Tag 2 = %d

%e formatta il primo tag tracciato come doppio valore a virgola mobile con un esponente e %d formatta il secondo tag tracciato come numero intero decimale con segno.

In questo caso, si ha un'istruzione di punto di tracciamento con due operandi Trace This (uno per REAL e uno per INT, sebbene il valore di qualsiasi tag possa essere formattato con qualsiasi indicatore).

La finestra del punto di tracciamento che ne risulta e che appare quando il punto di tracciamento viene attivata ha il seguente aspetto.


Il numero di slot indica lo slot che contiene il modulo emulatore che ha il punto di tracciamento o il breakpoint tracciato nella finestra.
L'intestazione (il testo che precede i due punti nella stringa del formato) appare qui.
Slot 1: My Tracepoint
Tag 1 = 6.488484e+002 and Tag 2 = 443
Tag 1 = 6.617994e+002 and Tag 2 = 447
Il testo per REAL (rappresentato nella stringa del formato come %e) compare qui.
Il testo per INT (rappresentato nella stringa del formato come %d) compare qui.
Freeze Display
Freeze Log
Clear Display

Indicatori di stato aritmetico: non influenzati

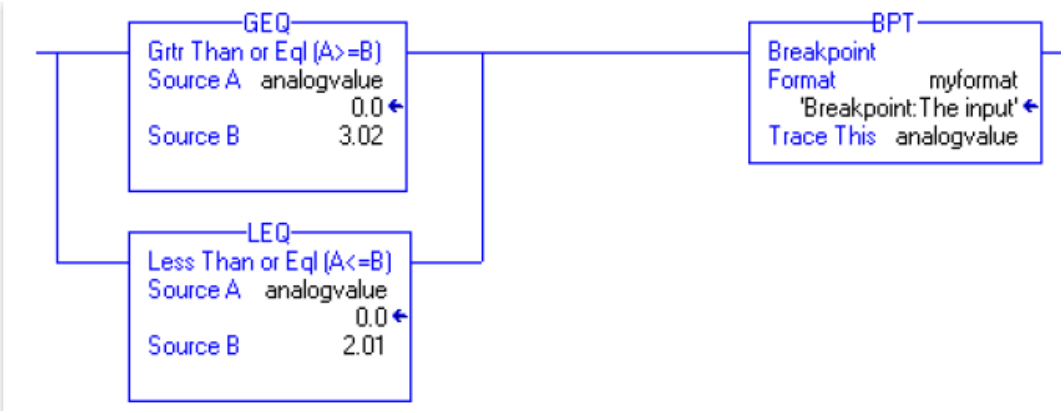
Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su vero. L'esecuzione salta al ramo che contiene l'istruzione LBL con il nome di etichetta di riferimento.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: È possibile visualizzare molti valori di tag con l'istruzione BPT. Tuttavia, la stringa di formattazione può contenere solo 82 caratteri. Poiché la stringa di formattazione richiede due caratteri per ogni tag che si desidera nel breakpoint non è possibile tracciare più di 41 tag con un'unica istruzione BPT. Tuttavia, per separare i dati dei tag nelle tracce, occorre comprendere spazi e ulteriore formattazione, riducendo così il numero di valori di tag che un'istruzione BPT è in grado di visualizzare a molto meno di 41.

Questo ramo mostra un breakpoint che interrompe l'esecuzione del programma quando un valore analogico è maggiore di 3,02 o inferiore di 2,01.

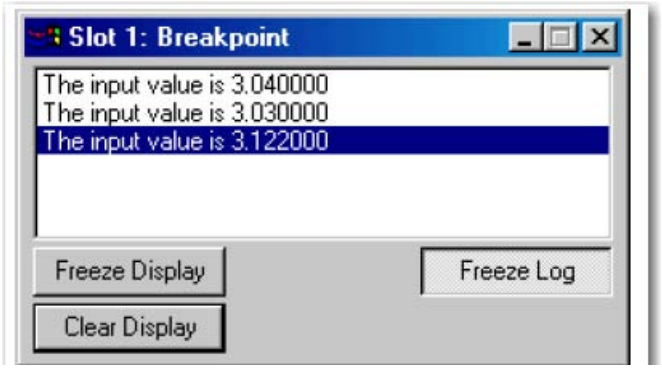


Si desidera visualizzare le informazioni sul breakpoint nella stringa Format (myformat). In questo caso, la stringa del formato contiene il testo seguente:

```
Breakpoint:The input value is %f
```

Quando il breakpoint viene attivato, la finestra di tracciamento del breakpoint mostra i caratteri prima dei due punti ("Breakpoint") nella barra del titolo della finestra. Gli altri caratteri compongono la traccia. In questo esempio, %f rappresenta il primo (e in questo caso l'unico) tag da tracciare ("analogvalue").

Le tracce risultati appaiono come segue.

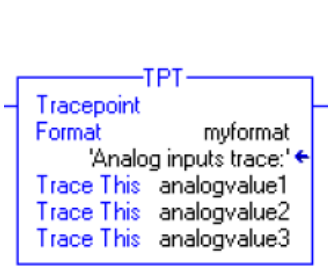


Tracepoints (TPT)

I punti di tracciamento registrano i dati selezionati quando un ramo è vero.

Operandi:

Ladder



Operando	Tipo	Formato	Descrizione
Format	stringa	tag	Stringa che imposta la formattazione per i report del tracciamento (su schermo e registrati nel disco).
Trace This	BOOL, SINT, INT, DINT, REAL	tag	Il tag che si desidera tracciare.

Descrizione:

I punti di tracciamento sono programmati con l'istruzione di uscita Tracepoint (TPT). Quando gli ingressi di un ramo contenente un'istruzione TPT sono veri, l'istruzione TPT scrive una voce di tracciamento nel display o nel file di registro del tracciamento.

È possibile tracciare molti tag con l'istruzione TPT. Tuttavia, la stringa di formattazione può contenere solo 82 caratteri. Poiché la stringa di formattazione richiede due caratteri per ogni tag che si desidera tracciare, non è possibile tracciare più di 41 tag con un'unica istruzione TPT. Tuttavia, per separare i dati dei tag nelle tracce, occorre comprendere spazi e ulteriore formattazione, riducendo così il numero di tag che un'istruzione TPT è in grado di tracciare a molto meno di 41.

Stringa Format

Con la stringa Format nelle istruzioni punto di tracciamento e breakpoint, è possibile controllare in che modo i tag tracciati appaiono nelle tracce o nelle finestre dei breakpoint. Il formato della stringa è mostrato di seguito:

intestazione:(testo)%(tipo)

dove *intestazione* è una stringa di testo che identifica il punto di tracciamento o il breakpoint, *testo* è una stringa che descrive il tag (o qualsiasi altro testo selezionato) e %(*tipo*) indica il formato del tag. Occorre un indicatore di tipo per ogni tag tracciato con l'istruzione punto di tracciamento o breakpoint.

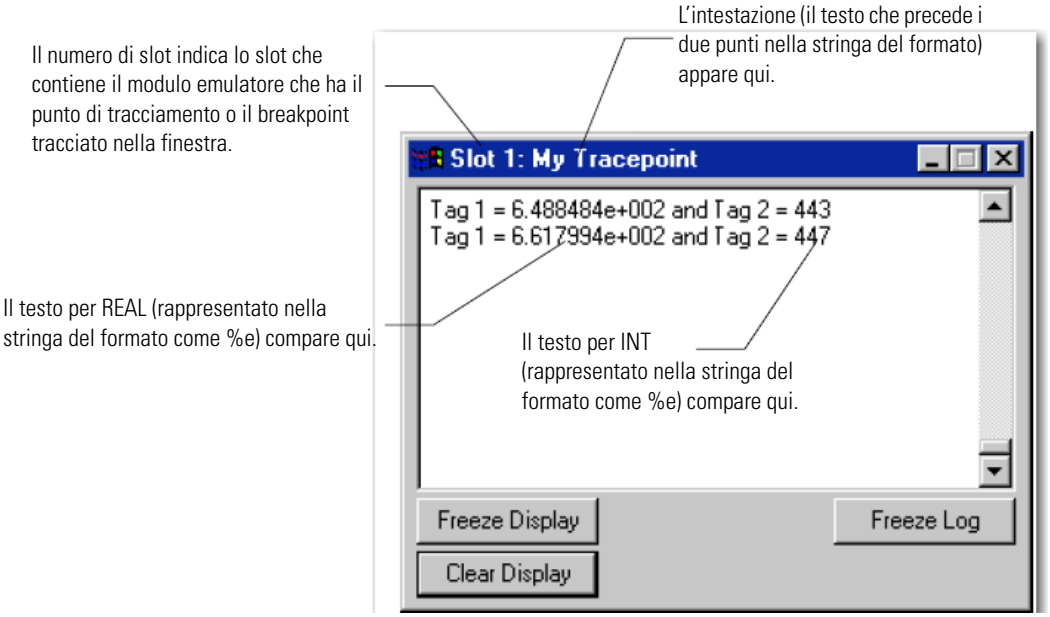
Ad esempio, è possibile formattare una stringa di un punto di tracciamento nel modo seguente:

My tracepoint:Tag 1 = %e e Tag 2 = %d

%e formatta il primo tag tracciato come doppio valore a virgola mobile con un esponente e %d formatta il secondo tag tracciato come numero intero decimale con segno.

In questo caso, si ha un'istruzione di punto di tracciamento con due operandi Trace This (uno per REAL e uno per INT, sebbene il valore di qualsiasi tag possa essere formattato con qualsiasi indicatore).

La finestra del punto di tracciamento che ne risulta e che appare quando il punto di tracciamento viene attivata ha il seguente aspetto.


Il numero di slot indica lo slot che contiene il modulo emulatore che ha il punto di tracciamento o il breakpoint tracciato nella finestra.
L'intestazione (il testo che precede i due punti nella stringa del formato) appare qui.
Slot 1: My Tracepoint
Tag 1 = 6.488484e+002 and Tag 2 = 443
Tag 1 = 6.617994e+002 and Tag 2 = 447
Il testo per REAL (rappresentato nella stringa del formato come %e) compare qui.
Il testo per INT (rappresentato nella stringa del formato come %e) compare qui.
Freeze Display
Freeze Log
Clear Display

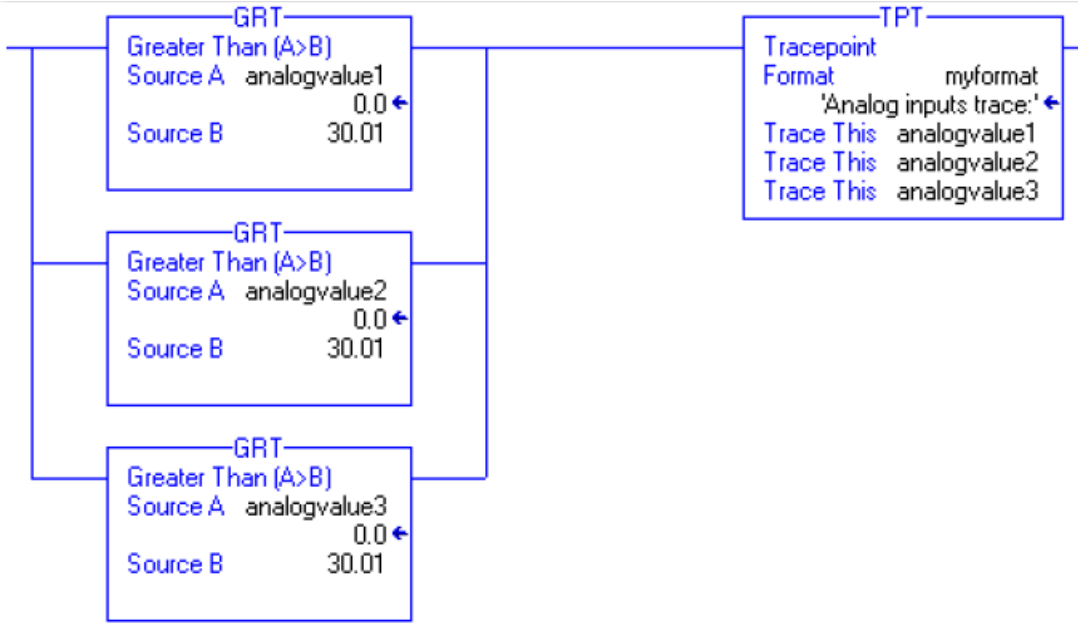
.Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esecuzione:

Condizione	Azione ladder
durante la prescansione	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è falsa	La condizione del ramo di uscita viene impostata su falso.
se la condizione del ramo di ingresso è vera	La condizione del ramo di uscita viene impostata su vero. L'esecuzione salta al ramo che contiene l'istruzione LBL con il nome di etichetta di riferimento.
post-scansione	La condizione del ramo di uscita viene impostata su falso.

Esempio: Questo ramo attiva una traccia di tre valori analogici quando uno di essi supera un dato valore (30,01).

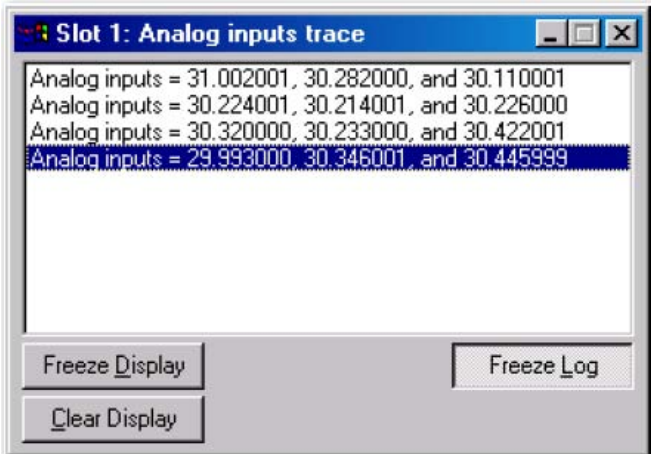


Si desidera visualizzare le informazioni sul punto di tracciamento nella stringa Format (myformat). In questo caso, la stringa del formato contiene il testo seguente:

```
Analog inputs trace:Analog inputs = %f, %f, and %f
```

Quando il punto di tracciamento viene attivato, la finestra di tracciamento del breakpoint mostra i caratteri prima dei due punti ("Analog inputs trace") nella barra del titolo. Gli altri caratteri compongono la traccia. In questo esempio, %f rappresenta i tag da tracciare ("analogvalue1," "analogvalue2," e "analogvalue3").

Le tracce risultati appaiono come segue.


Slot 1: Analog inputs trace
Analog inputs = 31.002001, 30.282000, and 30.110001
Analog inputs = 30.224001, 30.214001, and 30.226000
Analog inputs = 30.320000, 30.233000, and 30.422001
Analog inputs = 29.993000, 30.346001, and 30.445999
Freeze Display
Freeze Log
Clear Display

Quando questa traccia viene registrata nel disco, i caratteri prima dei due punti compaiono nelle tracce.

Questo indica quale punto di tracciamento ha provocato ogni voce di tracciamento. Questo è un esempio di voce di tracciamento. “Analog inputs trace:” è il testo di intestazione della stringa di formato del punto di tracciamento.

```
Analog inputs trace:Analog inputs = 31.00201, 30.282000,
e 30.110001
```

Attributi comuni

Introduzione

Questa appendice descrive gli attributi comuni alle istruzioni Logix.

Per informazioni su	Vedere pagina
Valori immediati	627
Conversioni di dati	627

Valori immediati

Ogni volta che si immette un valore immediato (costante) in formato decimale (ad es., –2, 3), il controllore memorizza il valore utilizzando 32 bit. Se si immette un valore in una radice non decimale, ad esempio in formato binario o esadecimale, e non si specificano tutti i 32 bit, il controllore inserisce uno zero nei bit non specificati (riempimento con zeri).

ESEMPIO Riempimento con zeri dei valori immediati

Se si immette	Il controllore memorizza
–1	16#ffff ffff (–1)
16#ffff (–1)	16#0000 ffff (65535)
8#1234 (668)	16#0000 029c (668)
2#1010 (10)	16#0000 000a (10)

Conversioni di dati

Le conversioni di dati hanno luogo quando nella propria programmazione si utilizzano tipi di dati misti:

Quando si programma in	Le conversioni possono avere luogo quando
Editor per la logica ladder	Si utilizzano tipi di dati misti per i parametri di un'istruzione
Blocco funzione	Vengono collegati due parametri aventi tipi di dati diversi

Le istruzioni vengono eseguite più rapidamente e richiedono meno memoria se tutti gli operandi dell'istruzione usano:

- Lo stesso tipo di dati
- Un tipo di dati ottimale:
  - Nella sezione "Operandi" di ciascuna istruzione contenuta in questo manuale, il tipo di dati ottimale viene riportato in **grassetto**.
  - DINT e REAL rappresentano di norma i tipi di dati ottimali.
  - La maggior parte delle istruzioni in blocco funzione supporta un solo tipo di dati (quello ottimale) per i propri operandi.

Se si mischiano tipi di dati diversi e si utilizzano tag di tipo non ottimale, il controllore converte i dati in base alle seguenti regole

- Uno *qualsiasi* degli operandi è un valore REAL?

Se	Gli operandi di ingresso (ad es., origine, tag di un'espressione, limite) vengono convertiti in
sì	REAL
no	DINT

- Al termine dell'esecuzione di un'istruzione, il risultato (un valore DINT o REAL) viene convertito nel tipo di dati della destinazione, se necessario.

Non è possibile specificare un tag BOOL in un'istruzione che opera su tipi di dati interi o REAL.

Poiché la conversione dei dati richiede ulteriore tempo e memoria, è possibile incrementare l'efficienza dei programmi come segue:

- utilizzando lo stesso tipo di dati nell'istruzione
- riducendo al minimo l'utilizzo dei tipi di dati SINT o INT

In sintesi, utilizzare nelle istruzioni tutti tag DINT o REAL insieme a valori immediati.

Le seguenti sezioni illustrano in che modo vengono convertiti i dati quando si utilizzano tag SINT o INT o tipi di dati misti.

Da SINT o INT a DINT

Per le istruzioni che convertono valori SINT o INT in valori DINT, il metodo di conversione viene descritto nelle sezioni “Operandi” del manuale.

Il metodo di conversione	Converte i dati inserendo
Estensione del segno	il valore del bit più a sinistra (il segno del valore) in ciascuna posizione bit alla sinistra dei bit esistenti fino a raggiungere 32 bit.
Riempimento con zeri	zeri alla sinistra dei bit esistenti fino a raggiungere 32 bit

Il seguente esempio mostra i risultati della conversione di un valore mediante estensione del segno e riempimento con zeri.

Il valore	2#1111_1111_1111_1111	(–1)
Viene convertito in questo valore mediante estensione del segno	2#1111_1111_1111_1111_1111_1111_1111_1111	(–1)
Viene convertito in questo valore mediante riempimento con zeri	2#0000_0000_0000_0000_1111_1111_1111_1111	(65535)

Dato che i valori immediati vengono sempre riempiti con zeri, la conversione di un valore SINT o INT *può* produrre risultati imprevisti. Nell’esempio seguente, il confronto è falso perché la sorgente A, un valore INT, viene convertito tramite un’estensione di segno, mentre la sorgente B, un valore immediato, è riempito con zeri.

Se si utilizza un tag SINT o INT e un valore immediato in un'istruzione che converte i dati mediante estensione del segno, usare uno di questi metodi per gestire i valori immediati:

- Specificare qualunque valore immediato nella radice decimale
- Se si immette il valore in una radice non decimale, specificare tutti i 32 bit del valore immediato. A tal fine, immettere il valore del bit più a sinistra in ogni posizione bit alla sua sinistra fino a raggiungere 32 bit.
- Creare un tag per ogni operando e utilizzare lo stesso tipo di dati in tutta l'istruzione. Per assegnare un valore costante, procedere in uno dei seguenti modi:
  - Immettere il valore costante in uno dei tag
  - Aggiungere un'istruzione MOV che sposti il valore in uno dei tag.
- Usare un'istruzione MEQ per verificare solo i bit richiesti

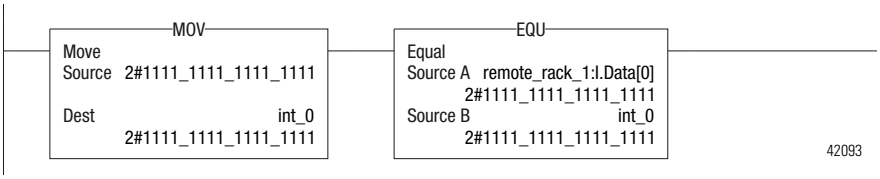
I seguenti esempi mostrano due metodi per combinare un valore immediato con un tag INT. In entrambi gli esempi vengono verificati i bit di un modulo I/O 1771 per stabilire se tutti i bit sono attivi. Poiché la parola dati di ingresso di un modulo I/O 1771 è un tag INT, è molto più semplice usare un valore costante a 16 bit.

ESEMPIO Utilizzo di un tag INT con un valore immediato

Dato che *remote_rack_1:I.Data[0]* è un tag INT, anche il valore di confronto viene immesso come tag INT.

ESEMPIO Utilizzo di un tag INT con un valore immediato

Dato che *remote_rack_1:I.Data[0]* è un tag INT, il valore di confronto viene prima spostato in *int_0*, anch'esso un tag INT. L'istruzione EQU confronterà quindi i due tag.



Da intero a REAL

Il controllore memorizza i valori REAL in formato numerico a virgola mobile a singola precisione IEEE. Viene utilizzato un bit per il segno del valore, 23 bit per il valore della base e otto bit per l'esponente (in totale 32 bit). Se in un'istruzione si utilizzano come ingressi un tag di interi (SINT, INT o DINT) e un tag REAL, il controllore converte il valore intero in valore REAL prima dell'esecuzione dell'istruzione stessa.

- Un valore SINT o INT viene sempre convertito nello stesso valore REAL.
- Un valore DINT può non essere convertito nello stesso valore REAL:
  - Un valore REAL usa fino a 24 bit per il valore della base (23 bit memorizzati più un bit "nascosto").
  - Un valore DINT usa fino a 32 bit per il valore (1 per il segno e 31 per il valore).
  - Se il valore DINT richiede più di 24 bit significativi, *potrebbe non* essere convertito nello stesso valore REAL. In questo caso, il controllore esegue un arrotondamento al valore REAL più vicino utilizzando 24 bit significativi.

Da DINT a SINT o INT

Per convertire un valore DINT in un valore SINT o INT, il controllore tronca la parte superiore del valore DINT e, se necessario, imposta l'indicatore di stato di overflow. Il seguente esempio mostra il risultato di una conversione da DINT a SINT o INT.

ESEMPIO Conversione di un valore DINT in un valore INT e SINT

Il valore DINT	Viene convertito nel valore inferiore	
16#0001_0081 (65.665)	INT:	16#0081 (129)
	SINT:	16#81 (–127)

Da REAL a intero

Per convertire un valore REAL in un valore intero, il controllore arrotonda la parte frazionaria e tronca la parte superiore della parte non frazionaria. In caso di perdita di dati, il controllore imposta l'indicatore di stato di overflow. I numeri vengono arrotondati come segue:

- I numeri diversi da *x*,5 vengono arrotondati al numero intero più vicino
- *X*,5 viene arrotondato al numero pari più vicino.

Il seguente esempio mostra il risultato della conversione di valori REAL in valori DINT.

ESEMPIO Conversione di valori REAL in valori DINT

Il valore REAL	Viene convertito nel valore DINT
–2,5	–2
–1,6	–2
–1,5	–2
–1,4	–1
1,4	1
1,5	2
1,6	2
2,5	2

IMPORTANTE Gli indicatori di stato aritmetico vengono impostati in base al valore memorizzato. Le istruzioni che di norma non influenzano le parole chiave di stato aritmetico potrebbero farlo se vi è una conversione di tipo a causa dell'utilizzo di tipi di dati misti per i parametri dell'istruzione. La conversione di tipo imposta le parole chiave di stato aritmetico.

Attributi di un blocco funzione

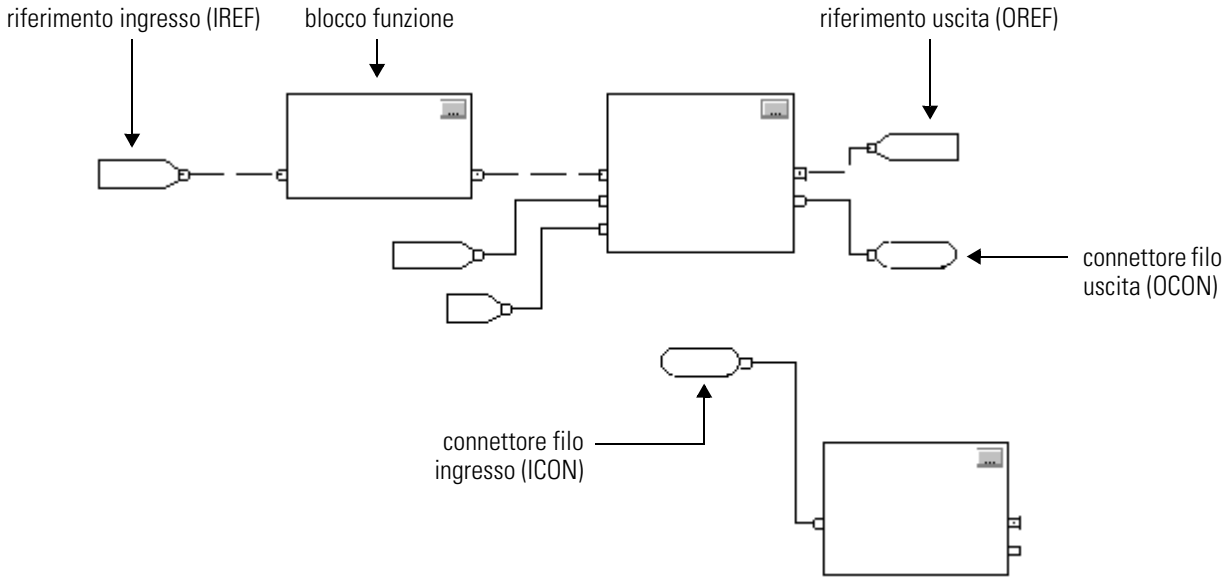
Introduzione

Questa appendice tratta aspetti riguardanti unicamente le istruzioni di un blocco funzione. Consultare le informazioni presenti in questa appendice per comprendere appieno la modalità di funzionamento delle routine del blocco funzione.

IMPORTANTE Quando si programma con blocchi funzione, limitare l'intervallo di unità ingegneristiche a $+/-10^{+/-15}$ dato che i calcoli interni a virgola mobile vengono eseguiti usando la virgola mobile a singola precisione. Le unità ingegneristiche che non rientrano in questo intervallo possono compromettere la precisione, se i risultati si avvicinano ai limiti della virgola mobile a singola precisione ($+/-10^{+/-38}$).

Selezione degli elementi del blocco funzione

Per controllare un dispositivo, utilizzare i seguenti elementi:



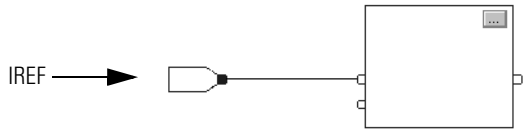
Usare la seguente tabella per selezionare gli elementi del blocco funzione:z

Se si desidera	Usare
fornire un valore da un dispositivo di ingresso o un tag	un riferimento ingresso (IREF)
inviare un valore a un dispositivo di uscita o un tag	un riferimento uscita (OREF)

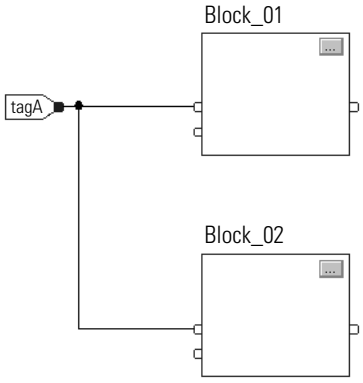
Se si desidera	Usare
eseguire un'operazione su valori di ingresso o produrre valori di uscita	un blocco funzione
trasferire dati tra blocchi funzioni quando sono: • lontani sulla stesso foglio • su fogli diversi nella stessa routine	un connettore filo di uscita (OCON) e connettore filo di ingresso (ICON)
disperdere i dati in diversi punti della routine	un connettore filo di uscita (OCON) e più connettori filo di ingresso (ICON)

Aggancio dati

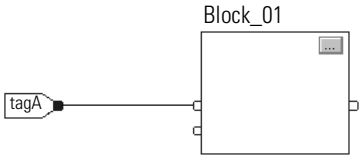
Se si utilizza un IREF per specificare i dati di ingresso di un'istruzione del blocco funzione, i dati contenuti nell'IREF vengono agganciati per la scansione della routine del blocco funzione. L'IREF aggancia dati dei tag di ambito programma e di ambito controllore. Il controllore aggiorna tutti i dati IREF all'inizio di ciascuna scansione.

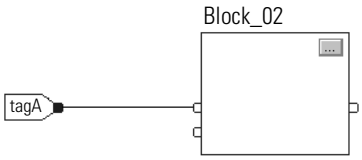


In questo esempio, il valore di tagA viene memorizzato all'inizio dell'esecuzione della routine. Il valore memorizzato viene usato nell'esecuzione di Block_01. Lo stesso valore memorizzato viene anche utilizzato nell'esecuzione di Block_02. Se il valore di tagA varia durante l'esecuzione della routine, il valore memorizzato di tagA nell'IREF rimane invariato fino alla successiva esecuzione della routine stessa.

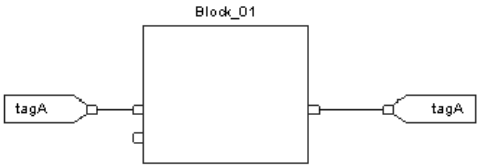


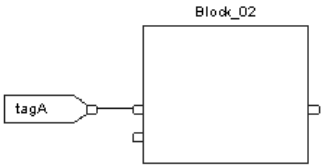
Questo esempio è uguale a quello riportato sopra. Il valore di tagA viene memorizzato solo una volta all'inizio dell'esecuzione della routine. La routine utilizza questo valore memorizzato per tutta l'esecuzione.





A partire dalla versione 11 del software RSLogix 5000, è possibile utilizzare lo stesso tag in più IREF e in un OREF nella stessa routine. Dato che i valori dei tag negli IREF vengono agganciati ad ogni scansione della routine, tutti gli IREF useranno lo stesso valore, anche se un OREF ottiene un valore di tag diverso durante l'esecuzione della routine. In questo esempio, se tagA ha un valore di 25,4 quando la routine avvia l'esecuzione di questa scansione, e Block_01 modifica il valore di tagA in 50,9, il secondo IREF collegato a Block_02 continuerà a usare il valore di 25,4 quando Block_02 eseguirà questa scansione. Il nuovo valore tagA di 50,9 non verrà usato da nessun IREF in questa routine fino all'avvio della scansione successiva.





Ordine di esecuzione

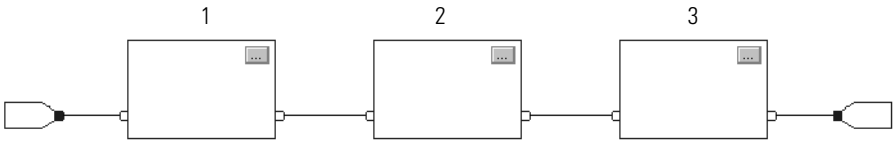
Il software di programmazione RSLogix 5000 determina automaticamente l'ordine di esecuzione dei blocchi funzione di una routine quando l'utente:

- verifica una routine in blocchi funzione
- verifica un progetto contenente una routine in blocchi funzione
- scarica un progetto contenente una routine in blocchi funzione

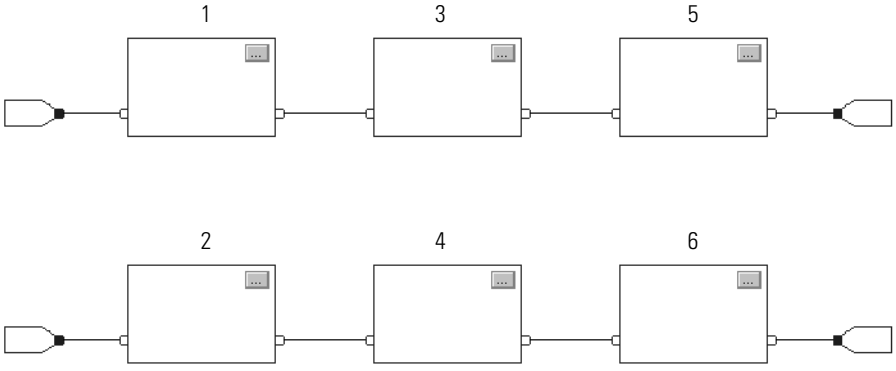
L'ordine di esecuzione viene definito cablando tra loro i blocchi funzione e indicando il flusso di dati su qualunque filo di feedback localizzato, se necessario.

Se i blocchi funzione non sono cablati tra loro, non è importante quale blocco viene eseguito per primo poiché non vi è alcun flusso di dati tra blocchi.

Se i blocchi vengono cablati in sequenza, l'ordine di esecuzione va dall'ingresso all'uscita. Gli ingressi di un blocco richiedono che i dati siano disponibili prima che il controllore possa eseguire il blocco stesso. Ad esempio, il blocco 2 deve essere eseguito prima del blocco 3, dato che le uscite del blocco 2 alimentano gli ingressi del blocco 3.

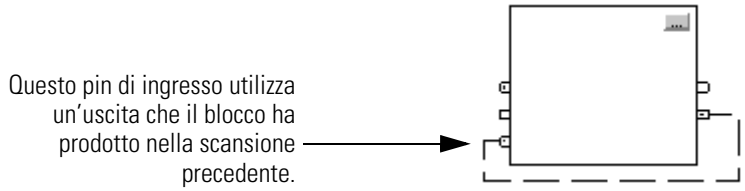


L'ordine di esecuzione si riferisce unicamente ai blocchi cablati tra loro. Il seguente esempio è particolarmente appropriato, dato che i due gruppi di blocchi non sono cablati tra loro. I blocchi di un gruppo specifico vengono eseguiti nell'ordine adeguato in relazione ai blocchi del gruppo stesso.

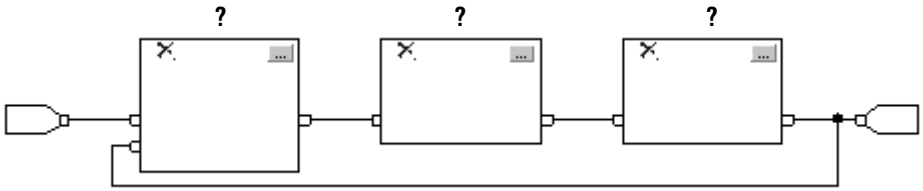


Risoluzione di un loop

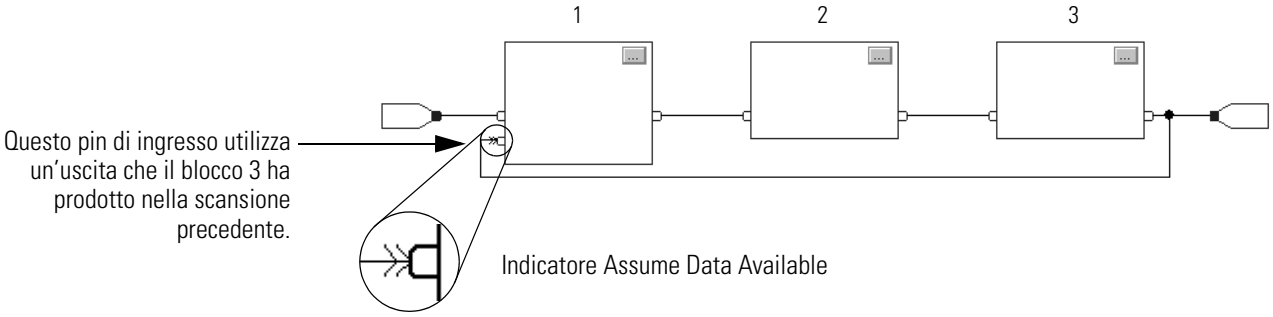
Per creare un loop di feedback intorno a un blocco, cablare un pin di uscita del blocco a un pin di ingresso dello stesso blocco. Il seguente esempio è corretto. Il loop contiene un unico blocco, pertanto l'ordine di esecuzione non è importante.



Se un gruppo di blocchi si trova in un loop, il controllore non è in grado di determinare quale blocco debba essere eseguito per primo. In altre parole, non è in grado di risolvere il loop.

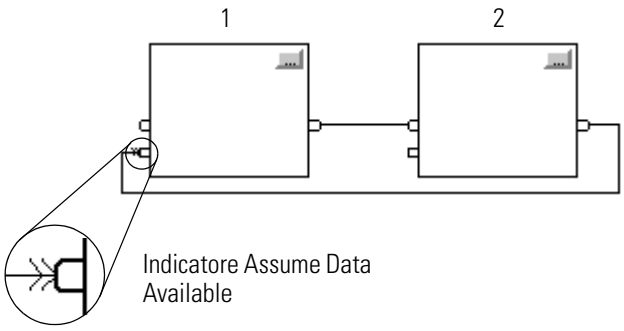
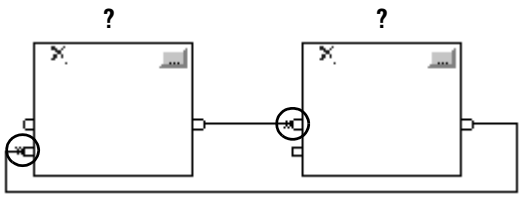


Per identificare quale blocco eseguire per primo, contrassegnare il filo di ingresso che crea il loop (il filo di feedback) con l'indicatore *Assume Data Available*. In questo esempio, il blocco 1 utilizza l'uscita del blocco 3 prodotta nella precedente esecuzione della routine.


1
2
3
Questo pin di ingresso utilizza un'uscita che il blocco 3 ha prodotto nella scansione precedente.
Indicatore Assume Data Available

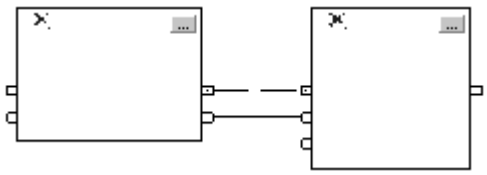
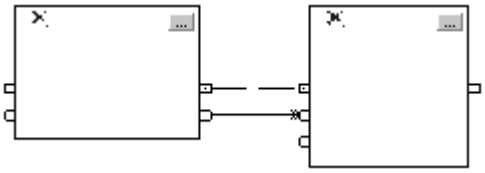
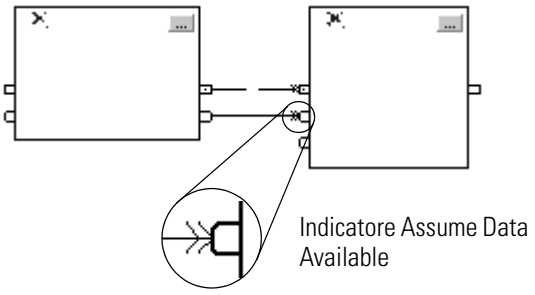
L'indicatore A*ssume Data Available* definisce il flusso di dati all'interno del loop. La freccia indica che i dati servono come dati in ingresso nel primo blocco del ciclo.

Non contrassegnare tutti i fili di un loop con l'indicatore *Assume Data Available*.

Quanto segue è corretto	Quanto segue non è corretto
 L'indicatore *Assume Data Available* definisce il flusso di dati all'interno del ciclo.	 Il controllore non è in grado di risolvere il loop perché tutti i fili usano l'indicatore *Assume Data Available*.

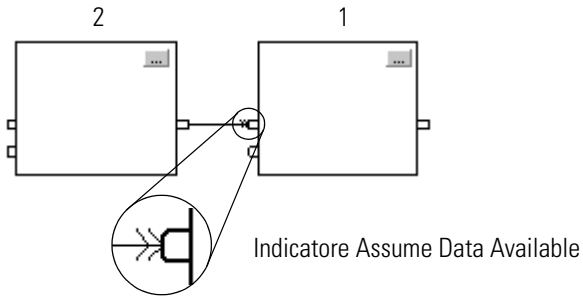
Risoluzione del flusso di dati tra due blocchi

Se si usano due o più fili per collegare due blocchi, usare gli stessi indicatori di flusso di dati per tutti i fili tra i due blocchi.

Quanto segue è corretto	Quanto segue non è corretto
 Nessun filo usa l'indicatore *Assume Data Available*.	 Un filo usa l'indicatore *Assume Data Available* e l'altro no.
 Entrambi i fili usano l'indicatore *Assume Data Available*.	

Creazione di un ritardo di una scansione

Per produrre un ritardo di una scansione tra blocchi, usare l'indicatore *Assume Data Available*. Nell'esempio seguente, il blocco 1 viene eseguito per primo. Utilizza l'uscita del blocco 2 che è stata prodotta nella scansione precedente della routine.



Riepilogo

In sintesi, una routine in blocchi funzione viene eseguita in questo ordine:

1. Il controllore aggancia tutti i valori dei dati contenuti negli IREF.
2. Il controllore esegue gli altri blocchi funzione nell'ordine determinato dalla modalità di cablaggio.
3. Il controllore scrive uscite negli OREF.

Risposte dei blocchi funzione a condizioni di overflow

In generale, le istruzioni in blocchi funzione che mantengono la cronologia non aggiornano la cronologia stessa con valori ±NAN o ±INF quando si verifica un overflow. Ogni istruzione ha una delle seguenti risposte a una condizione di overflow:

Risposta 1:		**Risposta 2:**	**Risposta 3:**	
I blocchi eseguono il proprio algoritmo e verificano eventuali valori nel risultato ±NAN o ±INF. Se ±NAN o ±INF, il blocco colloca in uscita ±NAN o ±INF.		Blocchi con limitazione dell'uscita eseguono il proprio algoritmo e verificano eventuali valori nel risultato ±NAN o ±INF. I limiti dell'uscita sono definiti dai parametri di ingresso HighLimit e LowLimit. Con ±INF, il blocco colloca in uscita un risultato limitato. Con ±NAN, i limiti dell'uscita non vengono usati e il blocco colloca in uscita ±NAN.	La condizione di overflow non è applicabile. Queste istruzioni hanno di norma un'uscita booleana.	
ALM	NTCH	HLL	BAND	OSRI
DEDT	PMUL	INTG	BNOT	RESD
DERV	POSP	PI	BOR	RTOR
ESEL	RLIM	PIDE	BXOR	SETD
FGEN	RMPS	SCL	CUTD	TOFR
HPF	SCRV	SOC	D2SD	TONR
LDL2	SEL		D3SD	
LDLG	SNEG		DFF	
LPF	SRTP		JKFF	
MAVE	SSUM		OSFI	
MAXC	TOT			
MINC	UPDN			
MSTD				
MUX				

Modalità di temporizzazione

Queste istruzioni di controllo processo e controllo assi supportano diverse modalità di temporizzazione.

DEDT	LDLG	RLIM
DERV	LPF	SCRV
HPF	NTCH	SOC
INTG	PI	TOT
LDL2	PIDE	

Vi sono tre diverse modalità di temporizzazione:

<table>
<tr><th>Modalità di temporizzazione</th><th>Descrizione</th></tr>
<tr><td>periodica</td><td>La modalità periodica è la modalità predefinita ed è adatta alla maggior parte delle applicazioni di controllo. Si consiglia di collocare le istruzioni che usano questa modalità in una routine eseguita in un task periodico. Il tempo delta (DeltaT) dell'istruzione è determinato come segue:
<table>
<tr><th>Se l'istruzione viene eseguita in</th><th>Delta è uguale a</th></tr>
<tr><td>Task periodico</td><td>Periodo del task</td></tr>
<tr><td>Evento o task continuo</td><td>Tempo trascorso dall'esecuzione precedente

Il controllore tronca il tempo trascorso ai millisecondi interi (ms). Ad esempio, se il tempo trascorso = 10,5 ms, il controllore imposta DeltaT = 10 ms.</td></tr>
<tr><td colspan="2">L'aggiornamento dell'ingresso del processo deve essere sincronizzato con l'esecuzione del task o campionato a una velocità di 5–10 volte superiore all'esecuzione del task stesso, al fine di ridurre al minimo l'errore di campionamento tra l'ingresso e l'istruzione.</td></tr>
</table></td></tr>
<tr><td>campionamento multiplo</td><td>Nella modalità di campionamento multiplo, il tempo delta (DeltaT) usato dall'istruzione è il valore scritto nel parametro OversampleDT dell'istruzione stessa. Se l'ingresso del processo ha un valore di registrazione cronologica, utilizzare la modalità di campionamento in tempo reale.

Aggiungere logica al proprio programma per controllare il momento di esecuzione dell'istruzione. Ad esempio, è possibile usare un temporizzatore impostato al valore OversampleDeltaT per controllare l'esecuzione usando l'ingresso EnableIn dell'istruzione.

L'ingresso del processo deve essere campionato a una velocità di 5–10 volte superiore all'esecuzione dell'istruzione, al fine di ridurre al minimo l'errore di campionamento tra l'ingresso e l'istruzione.</td></tr>
</table>

Modalità di temporizzazione	Descrizione
campionamento in tempo reale	Nella modalità di campionamento in tempo reale, il tempo delta (DeltaT) utilizzato dall'istruzione è la differenza tra due valori di registrazione cronologica corrispondenti agli aggiornamenti dell'ingresso del processo. Utilizzare questa modalità quando l'ingresso del processo ha una registrazione cronologica associata ai propri aggiornamenti e occorre un coordinamento preciso. Il valore di registrazione cronologica viene letto dal nome di tag immesso per il parametro RTSTimeStamp dell'istruzione. Di norma, questo nome di tag è un parametro del modulo di ingresso associato all'ingresso del processo. L'istruzione confronta il valore RTSTime configurato (tempo di aggiornamento previsto) con il DeltaT calcolato per stabilire se ogni aggiornamento dell'ingresso del processo viene letto dall'istruzione. Se DeltaT non rientra in 1 millisecondo del tempo di configurazione, l'istruzione imposta il bit di stato RTSMissed per indicare un problema di lettura degli aggiornamenti per l'ingresso del modulo.

Le istruzioni basate sul tempo richiedono un valore costante di DeltaT per consentire all'algoritmo di controllo di calcolare correttamente l'uscita del processo. Se DeltaT varia, vi sarà una discontinuità nell'uscita del processo. La gravità della discontinuità dipende dall'istruzione e dall'intervallo di variazione di DeltaT. La discontinuità si verifica se:

- l'istruzione non viene eseguita durante una scansione.
- l'istruzione viene eseguita più volte durante un task.
- il task è in esecuzione e la frequenza di scansione del task stesso o il tempo di campionamento dell'ingresso del processo varia.
- l'utente modifica la modalità della base tempo durante l'esecuzione del task.
- il parametro Order viene modificato su un blocco filtro durante l'esecuzione del task. Modificando il parametro Order, viene selezionato un algoritmo di controllo diverso all'interno dell'istruzione.

Parametri di istruzione comuni per le modalità di temporizzazione

Le istruzioni che supportano le modalità di base tempo presentano i seguenti parametri di ingresso e uscita:

Parametri di ingresso

Parametro di ingresso	Tipo di dati	Descrizione
TimingMode	DINT	Seleziona la modalità di esecuzione della temporizzazione. **Valore:** **Descrizione:** 0 modalità periodica 1 modalità di campionamento multiplo 2 modalità di campionamento in tempo reale valido = da 0 a 2 default = 0 Quando TimingMode = 0 e il task è periodico, viene abilitata la temporizzazione periodica e DeltaT viene impostato alla frequenza di scansione del task. Quando TimingMode = 0 e il task è continuo, viene abilitata la temporizzazione periodica e DeltaT corrisponde al tempo trascorso dall'ultima esecuzione dell'istruzione. Quando TimingMode = 1, viene abilitata la temporizzazione di campionamento multiplo e DeltaT viene impostato al valore del parametro OversampleDT. Quando TimingMode = 2, viene abilitata la temporizzazione di campionamento in tempo reale e DeltaT è la differenza tra il valore attuale e il valore precedente della registrazione cronologica letto tramite il modulo associato all'ingresso. Se TimingMode non è valido, l'istruzione imposta il bit appropriato in Status.
OversampleDT	REAL	Tempo di esecuzione per la temporizzazione di campionamento multiplo. Il valore usato per DeltaT è espresso in secondi. Se TimingMode = 1, allora OversampleDT = 0,0 disabilita l'esecuzione dell'algoritmo di controllo. Se il valore non è valido, l'istruzione imposta DeltaT = 0,0 e il bit appropriato in Status. valido = da 0 a 4194,303 secondi default = 0,0

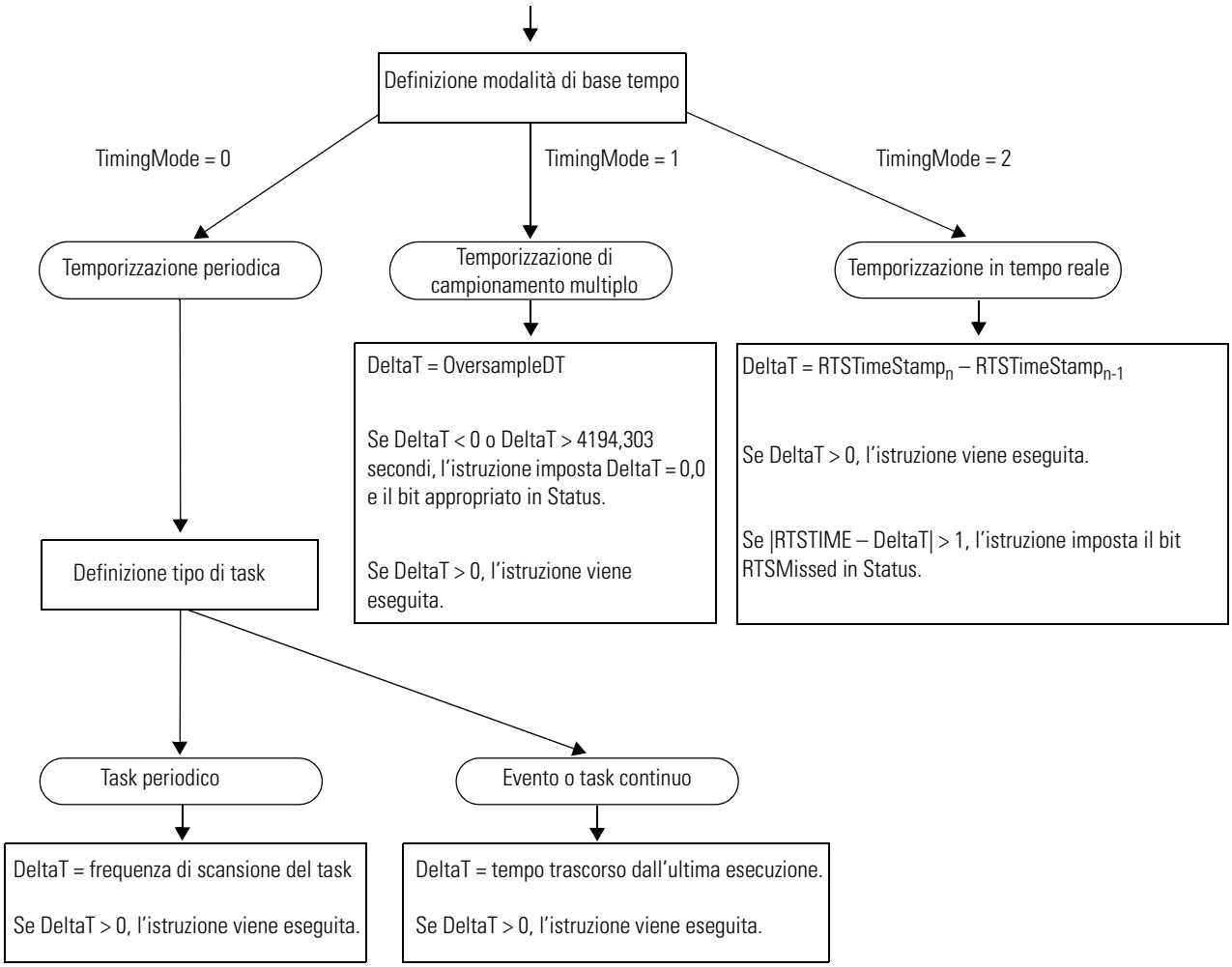
Parametro di ingresso	Tipo di dati	Descrizione
RTSTime	DINT	Tempo di aggiornamento del modulo per la temporizzazione di campionamento in tempo reale. Il tempo di aggiornamento DeltaT previsto è espresso in millisecondi. Il tempo di aggiornamento è di norma il valore utilizzato per configurare il tempo di aggiornamento del modulo. Se il valore non è valido, l'istruzione imposta il bit appropriato in Status e disabilita la verifica RTSMissed. valido = da 1 a 32 767 ms default = 1
RTSTimeStamp	DINT	Valore di registrazione cronologica del modulo per la temporizzazione di campionamento in tempo reale. Valore di registrazione cronologica corrispondente all'ultimo aggiornamento del segnale di ingresso. Questo valore viene usato per calcolare DeltaT. Se il valore non è valido, l'istruzione imposta l'appropriato bit in Status, disabilita l'esecuzione dell'algoritmo di controllo e la verifica RTSMissed. valido = da 1 a 32 767 ms (a capo da 32 767 a 0) 1 conteggio = 1 millisecondo default = 0

Parametri di uscita

Parametro di uscita	Tipo di dati	Descrizione
DeltaT	REAL	Tempo trascorso tra gli aggiornamenti. Si tratta del tempo trascorso, in secondi, usato dall'algoritmo di controllo per calcolare l'uscita del processo. Modalità periodica: DeltaT = frequenza di scansione del task se il task è periodico, DeltaT = tempo trascorso dall'ultima esecuzione dell'istruzione se il task è continuo Campionamento multiplo: DeltaT = OversampleDT Campionamento in tempo reale: DeltaT = ($RTSTimeStamp_n$ – $RTSTimeStamp_{n-1}$)
Status	DINT	Stato del blocco funzione.
TimingModeInv (Status.27)	BOOL	Valore TimingMode non valido.
RTSMissed (Status.28)	BOOL	Usato solo nella modalità di campionamento in tempo reale. Impostato quando ABS | DeltaT – RTSTime | > 1 (,001 secondi).
RTSTimeInv (Status.29)	BOOL	Valore RTSTime non valido.
RTSTimeStampInv (Status.30)	BOOL	Valore RTSTimeStamp non valido.
DeltaTInv (Status.31)	BOOL	Valore DeltaT non valido.

Prospetto delle modalità di temporizzazione

Lo schema seguente mostra in che modo un'istruzione determina la modalità di temporizzazione adeguata.


Definizione modalità di base tempo
TimingMode = 0
TimingMode = 1
TimingMode = 2
Temporizzazione periodica
Temporizzazione di campionamento multiplo
Temporizzazione in tempo reale
DeltaT = OversampleDT
Se DeltaT < 0 o DeltaT > 4194,303 secondi, l'istruzione imposta DeltaT = 0,0 e il bit appropriato in Status.
Se DeltaT > 0, l'istruzione viene eseguita.
DeltaT = RTSTimeStamp$_n$ – RTSTimeStamp$_{n-1}$
Se DeltaT > 0, l'istruzione viene eseguita.
Se |RTSTIME – DeltaT| > 1, l'istruzione imposta il bit RTSMissed in Status.
Definizione tipo di task
Task periodico
Evento o task continuo
DeltaT = frequenza di scansione del task
Se DeltaT > 0, l'istruzione viene eseguita.
DeltaT = tempo trascorso dall'ultima esecuzione.
Se DeltaT > 0, l'istruzione viene eseguita.

Controllo Programma/Operatore

Diverse istruzioni supportano il concetto di controllo Programma/Operatore, tra cui:

- Enhanced Select (ESEL)
- Totalizer (TOT)
- Enhanced PID (PIDE)
- Ramp/Soak (RMPS)
- Discrete 2-State Device (D2SD)
- Discrete 3-State Device (D3SD)

Il controllo Programma/Operatore consente di controllare queste istruzioni contemporaneamente sia dal programma utente che da un dispositivo di interfaccia operatore. Nel controllo Programma, l'istruzione viene controllata dagli ingressi Programma dell'istruzione stessa; nel controllo Operatore, l'istruzione viene controllata dagli ingressi Operatore dell'istruzione.

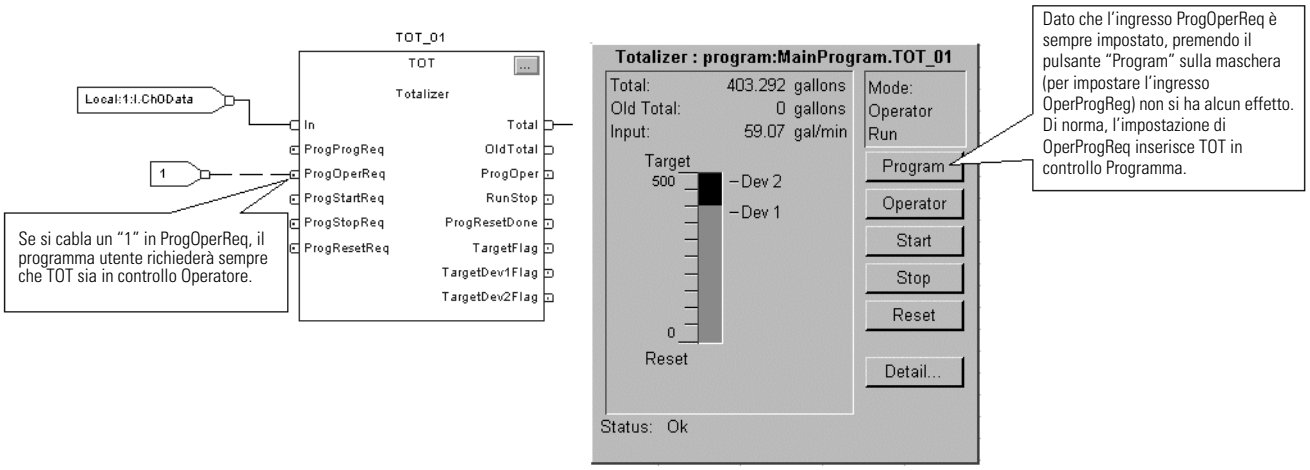
Il controllo Programma o Operatore viene stabilito usando i seguenti ingressi:

Ingresso	Descrizione
.ProgProgReq	Richiesta programma di passare al controllo Programma.
.ProgOperReq	Richiesta programma di passare al controllo Operatore.
.OperProgReq	Richiesta operatore di passare al controllo Programma.
.OperOperReq	Richiesta operatore di passare al controllo Operatore.

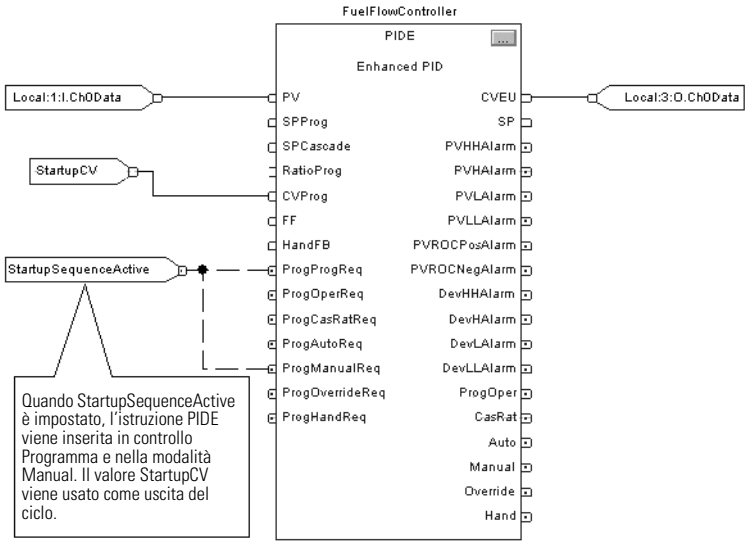
Per stabilire se un'istruzione si trova in controllo Programma o Operatore, esaminare l'uscita ProgOper. Se ProgOper è impostato, l'istruzione si trova in controllo Programma; se ProgOper è azzerato, l'istruzione si trova in controllo Operatore.

Se entrambi i bit di richiesta di ingresso sono impostati, il controllo Operatore ha la precedenza sul controllo Programma. Ad esempio, se ProgProgReq e ProgOperReq sono entrambi impostati, l'istruzione passa al controllo Operatore.

Gli ingressi di richiesta Programma hanno la precedenza sugli ingressi di richiesta Operatore. Ciò consente di usare gli ingressi ProgProgReq e ProgOperReq per “bloccare” un’istruzione nel controllo desiderato. Ad esempio, ipotizziamo che un’istruzione Totalizer venga sempre usata nel controllo Operatore e che il programma utente non controlli mai l’esecuzione o l’arresto del totalizzatore stesso. In questo caso, si potrebbe cablare un valore letterale di 1 in ProgOperReq. Ciò impedirebbe all’operatore di inserire il totalizzatore in controllo Programma impostando OperProgReq da un dispositivo di interfaccia operatore.



In maniera analoga, l'impostazione costante di ProgProgReq può "bloccare" l'istruzione in controllo Programma. Ciò è utile per sequenze di avvio automatiche in cui si desidera che il programma controlli l'azione dell'istruzione senza il rischio che un operatore prenda inavvertitamente il controllo dell'istruzione stessa. In questo esempio, il programma imposta l'ingresso ProgProgReq durante l'avvio, quindi lo azzera una volta che l'avvio è stato completato. Una volta azzerato l'ingresso ProgProgReq, l'istruzione rimane in controllo Programma finché non riceve una richiesta di modifica. Ad esempio, l'operatore potrebbe impostare l'ingresso OperOperReq da una maschera per assumere il controllo dell'istruzione. Il seguente esempio mostra come bloccare un'istruzione in controllo Programma.



Gli ingressi di richiesta operatore per un'istruzione vengono sempre azzerati dall'istruzione al momento dell'esecuzione. Ciò consente alle interfacce operatore di funzionare con queste istruzioni semplicemente impostando il bit di richiesta modalità desiderato. Non è necessario programmare l'interfaccia operatore affinché reimposti i bit di richiesta. Ad esempio, se un'interfaccia operatore imposta l'ingresso OperAutoReq per un'istruzione PIDE, al momento dell'esecuzione l'istruzione PIDE stabilisce la risposta appropriata e azzera OperAutoReq.

Gli ingressi di richiesta Programma non vengono solitamente azzerati dall'istruzione, essendo di norma cablati come ingressi nell'istruzione stessa. Se l'istruzione azzera questi ingressi, l'ingresso viene impostato di nuovo dall'ingresso cablato. Vi possono essere situazioni in cui si desidera utilizzare altra logica per impostare le richieste Programma in modo tale che vengano azzerate dall'istruzione. In questo caso, è possibile impostare l'ingresso ProgValueReset e l'istruzione azzererà sempre gli ingressi di richiesta di modalità Programmazione al momento dell'esecuzione.

In questo esempio viene usato un ramo di logica ladder di un'altra routine per agganciare con un singolo impulso un ProgAutoReq a un'istruzione PIDE premendo un pulsante. Poiché l'istruzione PIDE azzera automaticamente le richieste di modalità Programmazione, non è necessario scrivere logica ladder per azzerare ProgAutoReq dopo l'esecuzione della routine; l'istruzione PIDE riceverà solo una richiesta di passare ad Auto ogni volta che il pulsante viene premuto.

Quando si preme il pulsante TIC101AutoReq, agganciare con un singolo impulso ProgAutoReq all'istruzione PIDE TIC101. TIC101 è stato configurato con l'ingresso ProgValueReset impostato, pertanto l'istruzione PIDE azzera automaticamente ProgAutoReq al momento dell'esecuzione.

Note:

Programmazione in testo strutturato

Introduzione

Questa appendice descrive aspetti che riguardano unicamente la programmazione in testo strutturato. Consultare le informazioni presenti in questa appendice per comprendere appieno la modalità di esecuzione della programmazione in testo strutturato.

Per informazioni su	Vedere pagina
Sintassi del testo strutturato	651
Assegnazione	653
Espressioni	655
Istruzioni	662
Costrutti	663
Commenti	679

Sintassi del testo strutturato

Il testo strutturato è un linguaggio di programmazione testuale che utilizza istruzioni per definire l'azione da eseguire.

- Il testo strutturato non distingue tra maiuscole e minuscole.
- Utilizzare tabulazioni e ritorni a capo (righe separate) per rendere più semplice la lettura del testo strutturato. Le tabulazioni e i ritorni a capo non influenzano l'esecuzione del testo strutturato.

Il testo strutturato non distingue tra maiuscole e minuscole. Può contenere i seguenti componenti:

Termine	Definizione	Esempi
assegnazione (vedere pagina 653)	Usare un'assegnazione per assegnare valori ai tag. L'operatore := è l'operatore di assegnazione. Concludere l'assegnazione con un punto e virgola ";".	`tag := espressione;`

Termine	Definizione		Esempi
espressione (vedere pagina 655)	Un'espressione fa parte di un'istruzione completa di assegnazione o costrutto. Un'espressione restituisce un numero (espressione numerica) o uno stato vero o falso (espressione booleana).		
	Un'espressione contiene:		
	tag	Area definita della memoria in cui vengono memorizzati i dati (BOOL, SINT,INT,DINT, REAL, stringa).	`valore1`
	immediato	Valore costante.	`4`

	operatore	Simbolo o mnemonico che specifica un'operazione all'interno di un'espressione.	`tag1 + tag2`
			
			`tag1 >= valore1`

	funzione	Quando viene eseguita, una funzione restituisce un valore. Usare le parentesi per racchiudere l'operando di una funzione. Sebbene la loro sintassi sia simile, le funzioni differiscono dalle istruzioni, dato che le funzioni possono essere usate solo nelle espressioni. Le istruzioni non possono essere usate nelle espressioni.	`funzione(tag1)`

istruzione
 	
(vedere pagina 662)
	Un'istruzione è un enunciato indipendente.
	
	Un'istruzione utilizza le parentesi per racchiudere i propri operandi.
	
	A seconda dell'istruzione, vi possono essere uno, più operandi o nessuno.
	
	Quando viene eseguita, un'istruzione restituisce uno o più valori che fanno parte di una struttura di dati.
	
	Concludere l'istruzione con un punto e virgola ";".
	
	Sebbene la loro sintassi sia simile, le istruzioni differiscono dalle funzioni, dato che le istruzioni non possono essere usate nelle espressioni. Le funzioni possono essere usate solo nelle espressioni.
			`istruzione();`
			
			`istruzione(operando);`
			
			`istruzione(operando1, operando2,operando3);`

Termine	Definizione	Esempi

costrutto
 	
(vedere pagina 663)
	Istruzione condizionale usata per attivare un codice di testo strutturato (ovvero altre istruzioni).
	
	Concludere il costrutto con un punto e virgola ";".
		`IF...THEN`
		
		`CASE`
		
		`FOR...DO`
		
		`WHILE...DO`
		
		`REPEAT...UNTIL`
		
		`EXIT`

commento (vedere pagina 679)	Testo che spiega o chiarisce le azioni di una sezione di testo strutturato. • Usare i commenti per semplificare l'interpretazione del testo strutturato. • I commenti non influenzano l'esecuzione del testo strutturato. • I commenti possono apparire ovunque nel testo strutturato.	*`//commento`* `(*`*`inizio commento . . . fine commento`*`*)` *`/*inizio commento . . . fine commento*/`*

Assegnazione

Usare un'assegnazione per modificare il valore memorizzato in un tag. L'assegnazione ha la seguente sintassi:

`tag` := *`espressione`* ;

dove:

Componente	Descrizione	
`tag`	rappresenta il tag che riceve il nuovo valore il tag deve essere BOOL, SINT, INT, DINT o REAL	
:=	simbolo dell'assegnazione	
`espressione`	rappresenta il nuovo valore da assegnare al tag	
	Se *`tag`* è del tipo dati:	**Usare questo tipo di espressione:**
	BOOL	Espressione booleana
	SINT INT DINT REAL	Espressione numerica
;	Termina l'assegnazione	

Il *tag* conserva il valore assegnato finché questo non viene modificato da un'altra assegnazione.

L'espressione può essere semplice, come un valore immediato o un altro nome tag, oppure può essere complessa e comprendere diversi operatori e/o funzioni. Per ulteriori dettagli, vedere la sezione successiva "Espressioni" a pagina 655.

Specificare un'assegnazione non ritentiva

L'assegnazione non ritentiva è diversa dalla normale assegnazione descritta in precedenza, dato che il tag di un'assegnazione non ritentiva viene reimpostato a zero ogni volta che il controllore:

- entra in modalità Esecuzione
- abbandona il passo di un SFC se si configura l'SFC per l'*Automatic reset*. Ciò vale solo se si inserisce l'assegnazione nell'azione del passo o si utilizza l'azione per richiamare una routine di testo strutturato tramite un'istruzione JSR.

L'assegnazione non ritentiva presenta la seguente sintassi:

`tag` [:=] `espressione` ;

dove:

Componente	Descrizione	
`tag`	rappresenta il tag che riceve il nuovo valore il tag deve essere BOOL, SINT, INT, DINT o REAL	
[:=]	è il simbolo dell'assegnazione non ritentiva	
`espressione`	rappresenta il nuovo valore da assegnare al tag	
	Se `tag` è del tipo dati:	**Usare questo tipo di espressione:**
	BOOL	Espressione booleana
	SINT INT DINT REAL	Espressione numerica
;	Termina l'assegnazione	

Assegnare un carattere ASCII a una stringa

Usare l'operatore di assegnazione per assegnare un carattere ASCII a un elemento del membro DATA di un tag stringa. Per assegnare un carattere, specificare il valore del carattere o il nome tag, il membro DATA e l'elemento del carattere. Ad esempio:

Questo è corretto	Questo *non* è corretto
`string1.DATA[0]:= 65;`	`string1.DATA[0]:= A;`
`string1.DATA[0]:= string2.DATA[0];`	`string1 := string2;`

Per aggiungere o inserire una stringa di caratteri in un tag stringa, usare una delle seguenti istruzioni stringa ASCII:

Per	Utilizzare questa istruzione
aggiungere caratteri alla fine di una stringa	CONCAT
inserire caratteri in una stringa	INSERT

Espressioni

Un'espressione è un nome di tag, un'equazione o un confronto. Per scrivere un'espressione, utilizzare uno dei seguenti elementi:

- nome del tag che memorizza il valore (variabile)
- numero da immettere direttamente nell'espressione (valore immediato)
- funzioni, ad esempio: ABS, TRUNC
- operatori, ad esempio: +, –, <, >, And, Or

Nella scrittura delle espressioni, attenersi alle seguenti regole generali:

- Usare qualunque combinazione di caratteri maiuscoli e minuscoli. Ad esempio, sono accettabili queste tre varianti di "AND": AND, And, and.
- Per esigenze più complesse, usare le parentesi per raggruppare espressioni all'interno di espressioni. In questo modo l'espressione è più semplice da leggere e viene eseguita nella sequenza desiderata. Vedere "Determinare l'ordine di esecuzione" a pagina 661.

Nel testo strutturato si utilizzano due tipi di espressioni:

espressione booleana : Espressione che produce il valore BOOL di 1 (vero) o 0 (falso).

- Un'espressione booleana utilizza tag booleani, operatori relazionali e operatori logici per confrontare valori o verificare se le condizioni sono vere o false. Ad esempio, `tag1>65`.
- Un'espressione booleana semplice può essere composta da un unico tag booleano.
- Di norma, le espressioni booleane vengono usate per condizionare l'esecuzione di altra logica.

espressione numerica: Espressione che calcola un intero o un valore a virgola mobile.

- Un'espressione numerica utilizza operatori aritmetici, funzioni aritmetiche e operatori bit per bit. Ad esempio, `tag1+5`.
- L'espressione numerica viene spesso nidificata all'interno di un'espressione booleana. Ad esempio, `(tag1+5)>65`.

Utilizzare la seguente tabella per scegliere gli operatori per le espressioni:

Se si desidera	Allora
Calcolare un valore aritmetico	"Usare funzioni e operatori aritmetici" a pagina 657.
Confrontare due valori o stringhe	"Usare operatori relazionali" a pagina 658.
Verificare se le condizioni sono vere o false	"Usare operatori logici" a pagina 660.
Confrontare i bit nei valori	"Utilizzo di operatori bit per bit" a pagina 661.

Usare funzioni e operatori aritmetici

Nelle espressioni aritmetiche è possibile combinare più operatori e funzioni.

Gli operatori aritmetici calcolano i nuovi valori.

Per	Usare questo operatore	Tipo di dati ottimale
somma	+	DINT, REAL
sottrazione/negazione	–	DINT, REAL
moltiplicazione	*	DINT, REAL
esponente (x alla y)	**	DINT, REAL
divisione	/	DINT, REAL
modulo-divide	MOD	DINT, REAL

Le funzioni aritmetiche eseguono operazioni matematiche. Specificare una costante, un tag non booleano o un'espressione per la funzione.

Per	Usare questa funzione	Tipo di dati ottimale
valore assoluto	ABS *(espressione_numerica)*	DINT, REAL
arcocoseno	ACOS *(espressione_numerica)*	REAL
arcoseno	ASIN *(espressione_numerica)*	REAL
arcotangente	ATAN *(espressione_numerica)*	REAL
coseno	COS *(espressione_numerica)*	REAL
da radianti a gradi	DEG *(espressione_numerica)*	DINT, REAL
logaritmo naturale	LN *(espressione_numerica)*	REAL
logaritmo in base 10	LOG *(espressione_numerica)*	REAL
da gradi a radianti	RAD *(espressione_numerica)*	DINT, REAL
seno	SIN *(espressione_numerica)*	REAL
radice quadrata	SQRT *(espressione_numerica)*	DINT, REAL
tangente	TAN *(espressione_numerica)*	REAL
troncare	TRUNC *(espressione_numerica)*	DINT, REAL

Ad esempio:

Utilizzare questo formato	**ESEMPIO**	
	Per questa situazione	**Si scriverebbe**
valore1 operatore valore2	Se *gain_4* e *gain_4_adj* sono tag DINT e la specifica dell'utente indica: "Aggiungere 15 a *gain_4* e memorizzare il risultato in *gain_4_adj*."	`gain_4_adj := gain_4+15;`
operatore valore1	Se alarm e *high_alarm* sono tag DINT e la specifica dell'utente indica: "Negare *high_alarm* e memorizzare il risultato in *alarm*."	`alarm:= -high_alarm;`
funzione(espressione_numerica)	Se *overtravel* e *overtravel_POS* sono tag DINT e la specifica dell'utente indica: "Calcolare il valore assoluto di *overtravel* e memorizzare il risultato in *overtravel_POS*."	`overtravel_POS := ABS(overtravel);`
valore1 operatore funzione((valore2+valore3)/2)	Se *adjustment* e *position* sono tag DINT, *sensor1* e *sensor2* sono tag REAL e la specifica dell'utente indica: "Determinare il valore assoluto della media di *sensor1* e *sensor2*, aggiungere *adjustment* e memorizzare il risultato in *position*."	`position := adjustment + ABS((sensor1 + sensor2)/2);`

Usare operatori relazionali

Gli operatori relazionali confrontano due valori o stringhe per fornire un risultato vero o falso. Il risultato di un'operazione relazionale è un valore booleano:

Se il confronto è	**Il risultato è**
vero	1
falso	0

Utilizzare i seguenti operatori relazionali:

Per il confronto:	**Usare questo operatore**	**Tipo di dati ottimale**
uguale	=	DINT, REAL, stringa
minore di	<	DINT, REAL, stringa
minore di o uguale	<=	DINT, REAL, stringa
maggiore di	>	DINT, REAL, stringa
maggiore di o uguale a	>=	DINT, REAL, stringa
diverso	<>	DINT, REAL, stringa

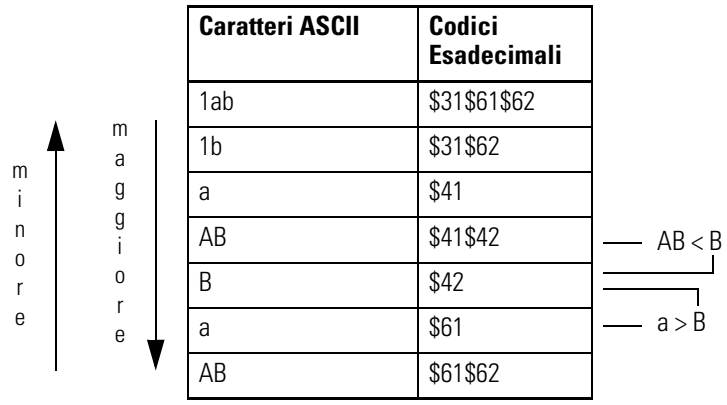
Ad esempio:

Utilizzare questo formato	**Esempio**	
	Per questa situazione	**Si scriverebbe**
valore1 operatore valore2	Se *temp* è un tag DINT e la specifica dell'utente indica: "Se *temp* è minore di 100°, allora…"	`IF temp<100 THEN...`
tagstringa1 operatore tagstringa2	Se *bar_code* e *dest* sono tag stringa e la specifica dell'utente indica: "Se *bar_code* è uguale a *dest*, allora…"	`IF bar_code=dest THEN...`
car1 operatore car2 Per immettere un carattere ASCII direttamente nell'espressione, inserire il valore decimale del carattere.	Se *bar_code* è un tag stringa e la specifica dell'utente indica: "Se *bar_code.DATA[0]* è uguale ad 'A', allora…"	`IF bar_code.DATA[0]=65 THEN...`
tag_booleano := espressioni_booleane	Se *count* e *length* sono tag DINT, *done* è un tag booleano e la specifica dell'utente indica: "Se *count* è maggiore di o uguale a *length*, il conteggio termina."	`done := (count >= length);`

Modalità di valutazione delle stringhe

I valori esadecimali dei caratteri ASCII determinano se una stringa è minore o maggiore di un'altra stringa.

- Quando le due stringhe sono ordinate come in un elenco telefonico, l'ordine delle stringhe determina quale è la maggiore.



Caratteri ASCII	**Codici Esadecimali**
1ab	$31$61$62
1b	$31$62
a	$41
AB	$41$42
B	$42
a	$61
AB	$61$62

- Le stringhe sono uguali se i loro caratteri corrispondono.
- I caratteri distinguono tra maiuscole e minuscole. "A" maiuscola ($41) non è uguale ad "a" minuscola ($61).

Per il valore decimale e il codice esadecimale di un carattere, vedere il retro copertina di questo manuale.

Usare operatori logici

Gli operatori logici consentono di verificare se più condizioni sono vere o false. Il risultato di un'operazione logica è un valore booleano:

Se il confronto è	Il risultato è
vero	1
falso	0

Utilizzare i seguenti operatori logici:

Per	Usare questo operatore	Tipo di dati
AND logico	&, AND	BOOL
OR logico	OR	BOOL
OR logico esclusivo	XOR	BOOL
complemento logico	NOT	BOOL

Ad esempio:

Utilizzare questo formato	ESEMPIO	
	Per questa situazione	**Si scriverebbe**
TagBooleano	Se *photoeye* è un tag booleano e la specifica dell'utente indica: "Se *photoeye* è attivo, allora…"	`IF photoeye THEN...`
NOT *TagBooleano*	Se *photoeye* è un tag booleano e la specifica dell'utente indica: "Se *photoeye* non è attivo, allora…"	`IF NOT photoeye THEN...`
espressione1 & *espressione2*	Se *photoeye* è un tag booleano, *temp* è un tag DINT e la specifica dell'utente indica: "Se *photoeye* è attivo e *temp* è minore di 100°, allora…".	`IF photoeye & (temp<100) THEN...`
espressione1 OR *espressione2*	Se *photoeye* è un tag booleano, *temp* è un tag DINT e la specifica dell'utente indica: "Se *photoeye* è attivo o *temp* è minore di 100°, allora…".	`IF photoeye OR (temp<100) THEN...`
espressione1 XOR *espressione2*	Se *photoeye1* e *photoeye2* sono tag booleani e la specifica dell'utente indica: "Se: • *photoeye1* è attivo mentre *photoeye2* non è attivo o • *photoeye1* non è attivo mentre *photoeye2* è attivo allora…"	`IF photoeye1 XOR photoeye2 THEN...`
TagBooleano := *espressione1* & *espressione2*	Se *photoeye1* e *photoeye2* sono tag booleani, *open* è un tag booleano e la specifica dell'utente indica: "Se *photoeye1* e *photoeye2* sono entrambi attivi, impostare *open* su vero".	`open := photoeye1 & photoeye2;`

Utilizzo di operatori bit per bit

Gli operatori bit per bit manipolano i bit di un valore in base a due valori.

Per	Usare questo operatore	Tipo di dati ottimale
AND di bit	&, AND	DINT
OR di bit	OR	DINT
OR di bit esclusivo	XOR	DINT
complemento di bit	NOT	DINT

Ad esempio:

Utilizzare questo formato	ESEMPIO	
	Per questa situazione	**Si scriverebbe**
valore1 operatore valore2	Se *input1*, *input2* e *result1* sono tag DINT e la specifica dell'utente indica: "Calcolare il risultato bit per bit di *input1* e *input2*. Memorizzare il risultato in *result1*."	`result1 := input1 AND input2;`

Determinare l'ordine di esecuzione

Le operazioni scritte in un'espressione vengono eseguite in un ordine definito, non necessariamente da sinistra a destra.

- Operazioni di ordine uguale vengono eseguite da sinistra a destra.
- Se un'espressione contiene più operatori o funzioni, raggruppare le condizioni tra parentesi "( )". Ciò assicura il corretto ordine di esecuzione e rende più semplice la lettura dell'espressione.

Ordine	Operazione
1.	( )
2.	funzione (…)
3.	**
4.	– (negare)
5.	NOT
6.	*, /, MOD
7.	+, – (sottrazione)
8.	<, <=, >, >=
9.	=, <>
10.	&, AND
11.	XOR
12.	OR

Istruzioni

Il testo strutturato può utilizzare anche le normali istruzioni. Consultare la Tabella indice all'inizio del manuale per un elenco delle istruzioni disponibili in testo strutturato. L'istruzione in testo strutturato viene eseguita a ogni scansione. Un'istruzione in testo strutturato all'interno di un costrutto viene eseguita ogni volta che le condizioni del costrutto sono vere. Se le condizioni del costrutto sono false, le istruzioni all'interno del costrutto non vengono sottoposte a scansione. Non vi è alcuna condizione di ramo o transizione di stato che attivi l'esecuzione.

Ciò differisce dalle istruzioni in blocco funzione che utilizzano EnableIn per attivare l'esecuzione. Le istruzioni in testo strutturato vengono eseguite come se EnableIn fosse sempre impostato.

Ciò differisce anche dalle istruzioni in logica ladder che utilizzano la condizione del ramo di ingresso per attivare l'esecuzione. Alcune istruzioni ladder vengono eseguite solo quando la condizione del ramo di ingresso passa da falsa a vera. Si tratta di istruzioni ladder di transizione. Nel testo strutturato, le istruzioni vengono eseguite ogni volta che vengono sottoposte a scansione, a meno che l'esecuzione non venga condizionata in precedenza.

Ad esempio, l'istruzione ABL è un'istruzione ladder di transizione. In questo esempio, l'istruzione ABL viene eseguita solo in una scansione in cui *tag_xic* passa da azzerato a impostato. L'istruzione ABL non viene eseguita quando *tag_xic* rimane impostato o quando *tag_xic* viene azzerato.

Nel testo strutturato, se si scrive questo esempio come:

```
IF tag_xic THEN ABL(0,serial_control);

END_IF;
```

l'istruzione ABL verrà eseguita ad ogni scansione in cui *tag_xic* è impostato, non solo quando *tag_xic* passa da azzerato a impostato.

Se si desidera che l'istruzione ABL venga eseguita solo quando *tag_xic* passa da azzerato a impostato, è necessario condizionare l'istruzione in testo strutturato. Usare un'istruzione One Shot per attivare l'esecuzione.

```
osri_1.InputBit := tag_xic;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
     ABL(0,serial_control);
END_IF;
```

Costrutti

I costrutti possono essere programmati come elementi indipendenti o nidificati all'interno di altri costrutti.

Se si desidera	Usare questo costrutto	Disponibile in questi linguaggi	Vedere pagina
eseguire un'azione se o quando si verificano condizioni specifiche	IF...THEN	testo strutturato	664
scegliere cosa fare in base a un valore numerico	CASE...OF	testo strutturato	667
eseguire un'azione un numero specifico di volte prima di eseguire qualunque altra azione	FOR...DO	testo strutturato	670
continuare a eseguire un'azione finché alcune condizioni sono vere	WHILE...DO	testo strutturato	673
continuare a eseguire un'azione finché una condizione non diventa vera	REPEAT...UNTIL	testo strutturato	676

Alcune parole chiave sono riservate all'uso futuro

Questi costrutti non sono disponibili:

- GOTO
- REPEAT

Il software RSLogix 5000 non consente di utilizzarli come nomi di tag né costrutti.

IF...THEN

Usare IF…THEN per eseguire un'azione se o quando si verificano condizioni specifiche.

Operandi:

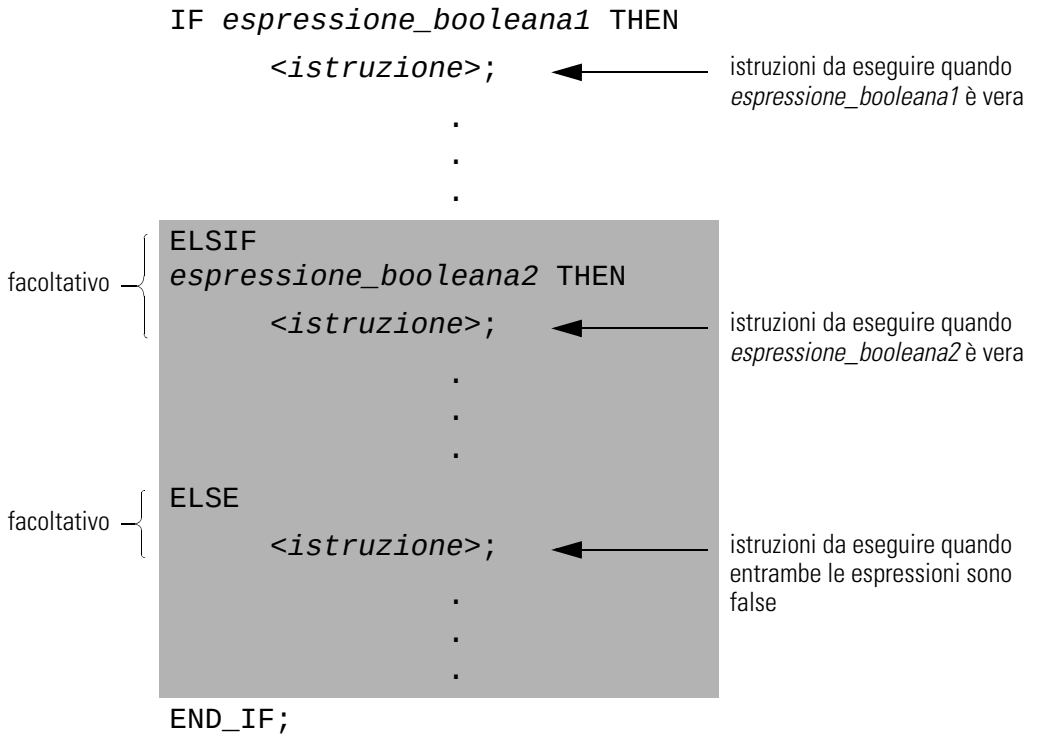
Testo strutturato

```
IF espressione_booleana
THEN
<istruzione>;
END_IF;
```

Operando	Tipo	Formato	Digitare
espressione_ booleana	BOOL	tag espressione	Tag booleano o espressione booleana che restituisce un valore booleano (espressione booleana)

Descrizione:

La sintassi è:



Per usare ELSIF o ELSE, seguire queste linee guida:

1. Per operare una scelta tra diversi gruppi di istruzioni possibili, aggiungere una o più istruzioni ELSIF.
   - Ogni ELSIF rappresenta un percorso alternativo.
   - Specificare il numero di percorsi ELSIF necessario.
   - Il controllore esegue il primo IF o ELSIF vero e ignora il resto delle istruzioni ELSIF ed ELSE.
2. Per eseguire un'azione quando tutte le condizioni IF o ELSIF sono false, aggiungere un'istruzione ELSE.

La seguente tabella presenta una sintesi di diverse combinazioni di IF, THEN, ELSIF ed ELSE.

Se si desidera	E	Usare questo costrutto
eseguire un'azione se o quando le condizioni sono vere	non eseguire alcuna azione se le condizioni sono false	IF…THEN
	eseguire un'azione diversa se le condizioni sono false	IF…THEN…ELSE
scegliere tra istruzioni alternative (o gruppi di istruzioni) in base alle condizioni di ingresso	non eseguire alcuna azione se le condizioni sono false	IF…THEN…ELSIF
	assegnare istruzioni predefinite se tutte le condizioni sono false	IF…THEN…ELSIF…ELSE

Indicatori di stato aritmetico non influenzati

Condizioni di errore: nessuna

Esempio 1: IF…THEN

Per	Immettere questo testo strutturato
SE scarti > 3 allora	`IF scarti > 3 THEN`
trasportatore = off (0)	`    trasportatore:= 0;`
allarme = on (1)	`    allarme:= 1;`
	`END_IF;`

Esempio 2: IF…THEN…ELSE

Per	Immettere questo testo strutturato
Se contatto di direzione nastro = avanti (1) allora	`IF direzione_nastro THEN`
spia = off	`    spia := 0;`
Altrimenti spia = on	`ELSE`
	`    spia [:=] 1;`
	`END_IF;`

[:=] indica al controllore di azzerare *spia* quando il controllore:

- entra in modalità Run
- abbandona il passo di un SFC se si configura l'SFC per il *reset automatico*. Ciò vale solo se si inserisce l'assegnazione nell'azione del passo o si utilizza l'azione per richiamare una routine di testo strutturato tramite un'istruzione JSR.

Esempio 3: IF…THEN…ELSIF

Per	Immettere questo testo strutturato
Se finecorsa livello minimo zucchero = basso (on) e finecorsa livello massimo zucchero = non alto (on), allora	`IF Zucchero.Basso & Zucchero.Alto THEN`
valvola ingresso = aperta (on)	`Valvola.Zucchero[:=] 1;`
Finché finecorsa livello massimo zucchero = alto (off)	`ELSIF NOT(Zucchero.Alto) THEN`
	`Valvola.Zucchero:= 0;`
	`END_IF;`

[:=] indica al controllore di azzerare *Valvola.Zucchero* quando il controllore:

- entra in modalità RUN
- abbandona il passo di un SFC se si configura l'SFC per il *reset automatico*. Ciò vale solo se si inserisce l'assegnazione nell'azione del passo o si utilizza l'azione per richiamare una routine di testo strutturato tramite un'istruzione JSR.

Esempio 4: IF…THEN…ELSIF…ELSE

Per	Immettere questo testo strutturato
Se temperatura serbatoio > 100	`IF temp.serbatoio > 200 THEN`
allora pompa = lenta	`pompa.veloce :=1; pompa.lenta :=0; pompa.off :=0;`
Se temperatura serbatoio > 200	`ELSIF temp.serbatoio> 100 THEN`
allora pompa = veloce	`pompa.veloce :=0; pompa.lenta :=1; pompa.off :=0;`
altrimenti pompa = off	`ELSE`
	`pompa.veloce :=0; pompa.lenta :=0; pompa.off :=1;`
	`END_IF;`

CASE...OF

Usare CASE per scegliere quale azione eseguire in base a un valore numerico.

Operandi:

```
CASE espressione_numerica
OF
     selettore1: istruzione;
     selettoreN: istruzione;
ELSE
     istruzione;
END_CASE;
```

Testo strutturato

Operando	Tipo	Formato	Digitare
espressione_numerica	SINT	tag	tag o espressione che restituisce un numero (espressione numerica)
	INT	espressione	
	DINT		
	REAL		
selettore	SINT	immediato	stesso tipo di *espressione_numerica*
	INT		
	DINT		
	REAL		

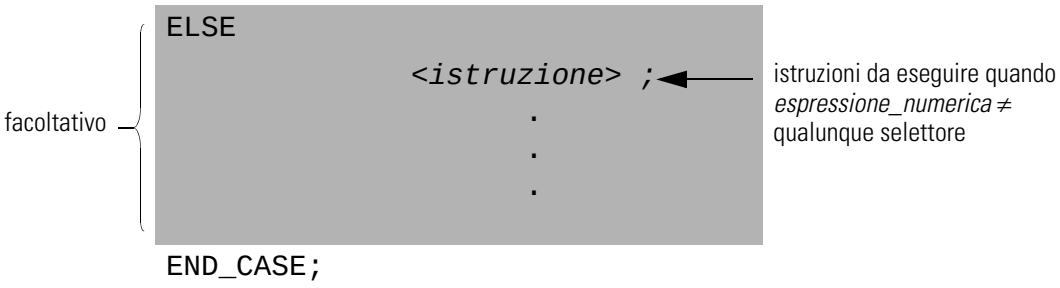
IMPORTANTE Se si utilizzano valori REAL, usare un intervallo di valori per un selettore, dato che un valore REAL ha più probabilità di rientrare in un intervallo di valori rispetto a un unico valore esattamente corrispondente a un valore specifico.

Descrizione:

La sintassi è:

```
CASE espressione_numerica OF
    selettore 1 :   <istruzione>;   <-- istruzioni da eseguire quando espressione_numerica = selettore1
                        .
                        .
                        .
    selettore 2 :   <istruzione>;   <-- istruzioni da eseguire quando espressione_numerica = selettore2
                        .
                        .
                        .
    selettore 3 :   <istruzione>;   <-- istruzioni da eseguire quando espressione_numerica = selettore3
                        .
                        .
                        .
```

specificare tutti i valori di selettore alternativi (percorsi) necessari



END_CASE;

Consultare la tabella alla pagina seguente per un elenco di valori di selettore validi.

La sintassi per l'immissione di valori di selettore è la seguente:

Quando il selettore è	Digitare
un valore	*valore: istruzione*
più valori distinti	*valore1*, *valore2*, *valoreN* : <*istruzione*> Usare una virgola (,) per separare ciascun valore.
un intervallo di valori	*valore1..valoreN* : <*istruzione*> Usare due puntini di sospensione (..) per identificare l'intervallo.
valori distinti più un intervallo di valori	*valorea, valoreb, valore1..valoreN* : <*istruzione*>

Il costrutto CASE è simile a un'istruzione di commutazione nei linguaggi di programmazione C o C++. Tuttavia, con il costrutto CASE il controllore esegue solo le istruzioni associate al primo valore corrispondente del selettore. L'esecuzione viene sempre interrotta dopo le istruzioni di quel selettore e passa all'istruzione END_CASE.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore: nessuna

Esempio:

Per	Immettere questo testo strutturato
Se numero ricetta = 1 allora	`CASE numero_ricetta OF`
Ingrediente A apertura 1 = aperta (1)	`1: Ingrediente_A.Apertura_1 :=1;`
Ingrediente B apertura 4 = aperta (1)	`Ingrediente_B.Apertura_4 :=1;`
Se numero ricetta = 2 o 3 allora	`2,3: Ingrediente_A.Apertura_4 :=1;`
Ingrediente A apertura 4 = aperta (1)	`Ingrediente_B.Apertura_2 :=1;`
Ingrediente B apertura 2 = aperta (1)	
Se numero ricetta = 4, 5, 6 o 7 allora	`4..7: Ingrediente_A.Apertura_4 :=1;`
Ingrediente A apertura 4 = aperta (1)	`Ingrediente_B.Apertura_2 :=1;`
Ingrediente B apertura 2 = aperta (1)	
Se numero ricetta = 8, 11, 12 o 13 allora	`8,11..13 Ingrediente_A.Apertura_1 :=1;`
Ingrediente A apertura 1 = aperta (1)	`Ingrediente_B.Apertura_4 :=1;`
Ingrediente B apertura 4 = aperta (1)	
Altrimenti tutte le aperture = chiuse (0)	`ELSE`
	`Ingrediente_A.Apertura_1 [:=]0;`
	`Ingrediente_A.Apertura_4 [:=]0;`
	`Ingrediente_B.Apertura_2 [:=]0;`
	`Ingrediente_B.Apertura_4 [:=]0;`
	`END_CASE;`

[:=] indica al controllore di azzerare anche i tag “apertura” quando il controllore:

- entra in modalità RUN
- abbandona il passo di un SFC se si configura l’SFC per il *reset automatico*. Ciò vale solo se si inserisce l’assegnazione nell’azione del passo o si utilizza l’azione per richiamare una routine di testo strutturato tramite un’istruzione JSR.

FOR...DO

Usare il ciclo FOR…DO per eseguire un'azione un numero specifico di volte prima di eseguire un'altra azione.

Operandi:

```
FOR conteggio:=
valore_iniziale TO
valore_finale BY incremento DO
    <istruzione>;
END_FOR;
```

Testo strutturato

Operando	Tipo	Formato	Descrizione
count	SINT INT DINT	tag	tag per la memorizzazione della posizione di conteggio nell'esecuzione di FOR…DO
initial_value	SINT INT DINT	tag espressione immediato	deve restituire un numero specifica il valore iniziale per il conteggio
final_value	SINT INT DINT	tag espressione immediato	specifica il valore finale per il conteggio; determina quando uscire dal ciclo
increment	SINT INT DINT	tag espressione immediato	(*facoltativo*) valore di incremento del conteggio per ogni ciclo Se non si specifica un incremento, il conteggio viene incrementato di 1.

IMPORTANTE

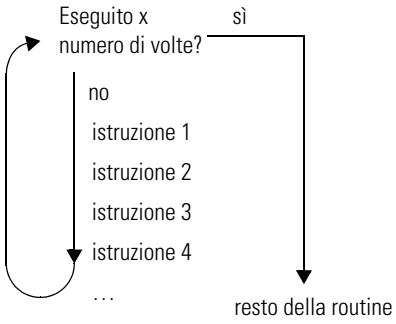
Non creare un numero eccessivo di iterazioni all'interno del ciclo in una singola scansione.

- Il controllore *non* esegue altre istruzioni della routine finché non completa il ciclo.
- Se il tempo necessario per completare il ciclo è maggiore del timer di watchdog per l'attività, si verificherà un errore grave.
- Provare a utilizzare un costrutto diverso, ad esempio IF...THEN.

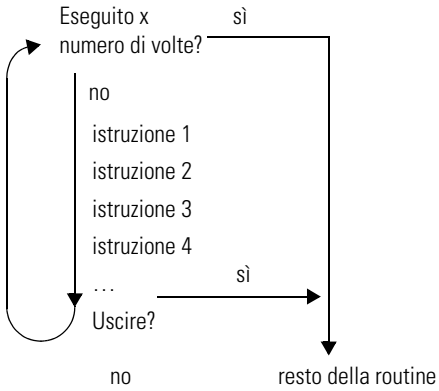
Descrizione: La sintassi è:

```
FOR conteggio := valore_ iniziale
   TO valore_finale
facoltativo { BY incremento                    Se non si specifica un incremento, il ciclo
                                               viene incrementato di 1.
   DO
   <istruzione>;
   IF espressione_booleana
   THEN
facoltativo      EXIT;                       <-- Se vi sono condizioni in cui si desidera
   END_IF;                                       uscire anticipatamente dal ciclo, usare
                                                 altre istruzioni, ad esempio un costrutto
                                                 IF...THEN, per condizionare un'istruzione
                                                 EXIT.
END_FOR;
```

I seguenti diagrammi mostrano come viene eseguito il ciclo FOR...DO e come un'istruzione EXIT produce un'uscita anticipata dal ciclo.



Il ciclo FOR…DO viene eseguito un numero specifico di volte.



Per arrestare il ciclo prima che il conteggio raggiunga l'ultimo valore, utilizzare un'istruzione EXIT.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
il ciclo del costrutto è troppo lungo	6	1

Esempio 1:

Per	Immettere questo testo strutturato

Azzerare i bit 0–31 in una matrice di BOOL:
 	
1. Inizializzare il tag *subscript* a 0.
 	
2. Azzerare *array[subscript]*. Ad esempio, quando *subscript* = 5, azzerare *array[5]*.
 	
3. Aggiungere 1 a *subscript*.
 	
4. Se *subscript* è ≤ di 31, ripetere 2 e 3.
 	
Altrimenti, arrestare.
	`For subscript:=0 to 31 by 1 do`
	`  array[subscript] := 0;`
	`End_for;`

Esempio 2:

Per	Immettere questo testo strutturato

Un tipo di dati definito dall'utente (struttura) memorizza le seguenti informazioni su un articolo dell'inventario
 	
• ID del codice a barre dell'articolo (tipo dati stringa)
• Quantità dell'articolo in magazzino (tipo dati DINT)
 	
Una matrice della struttura descritta sopra contiene un elemento per ogni articolo dell'inventario. Si desidera cercare un prodotto specifico nella matrice (utilizzare il suo codice a barre) e determinare la quantità in magazzino.
 	
1. Ottenere le dimensioni (numero di articoli) della matrice *Inventory* (inventario) e memorizzare il risultato in *Inventory_Items* (articoli inventario) (tag DINT).
 	
2. Inizializzare il tag *position* a 0.
 	
3. Se *Barcode* corrisponde all'ID di un articolo nella matrice, allora:
 	
a. Impostare il tag *Quantity* tag = *Inventory[position].Qty*. In questo modo si avrà la quantità dell'articolo presente in magazzino.
 	
b. Fine.
 	
Barcode è un tag stringa che memorizza il codice a barre dell'articolo cercato. Ad esempio, quando *position* = 5, confrontare *Barcode* con *Inventory[5].ID*.
 	
4. Aggiungere 1 a *position*.
 	
5. Se *position* è ≤ di (*Inventory_Items –1*), ripetere 3 e 4. Poiché i numeri di elemento iniziano a 0, l'ultimo elemento equivale al numero di elementi nella matrice meno 1.
 	
Altrimenti, arrestare.
	`SIZE(Inventory,0,Inventory_Items);`
	`For position:=0 to Inventory_Items – 1 do`
	`  If Barcode = Inventory[position].ID then`
	`  Quantity := Inventory[position].Qty;`
	`  Exit;`
	`  End_if;`
	`End_for;`

WHILE...DO

Utilizzare WHILE…DO per continuare a eseguire un'azione finché alcune condizioni sono vere.

Operandi:

Testo strutturato

```
WHILE espressione_booleana DO
    <istruzione>;
END_WHILE;
```

Operando	Tipo	Formato	Digitare
espressione_ booleana	BOOL	tag espressione	Tag booleano o espressione booleana che restituisce un valore booleano

IMPORTANTE

Non creare un numero eccessivo di iterazioni all'interno del ciclo in una singola scansione.

- Il controllore *non* esegue altre istruzioni della routine finché non completa il ciclo.
- Se il tempo necessario per completare il ciclo è maggiore del timer di watchdog per l'attività, si verificherà un errore grave.
- Provare a utilizzare un costrutto diverso, ad esempio IF...THEN.

Descrizione: La sintassi è:

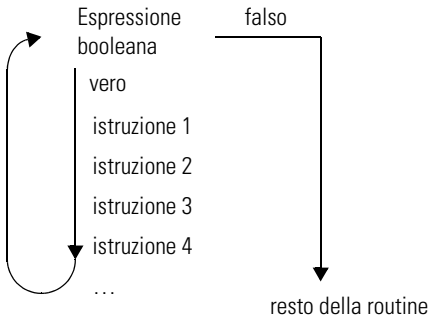
```
WHILE espressione_booleana1 DO
   <istruzione>;
   IF espressione_booleana2 THEN
     EXIT;
   END_IF;
END_WHILE;
```

istruzioni da eseguire mentre *espressione_booleana1* è vera

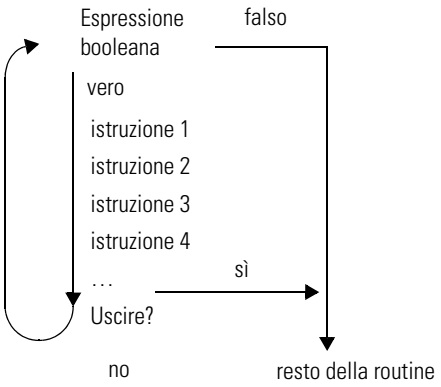
facoltativo

Se vi sono condizioni in cui si desidera uscire anticipatamente dal ciclo, usare altre istruzioni, ad esempio un costrutto IF...THEN, per condizionare un'istruzione EXIT.

I seguenti diagrammi mostrano come viene eseguito il ciclo WHILE...DO e come un'istruzione EXIT produce un'uscita anticipata dal ciclo.



Mentre *espressione_booleana* è vera, il controllore esegue solo le istruzioni all'interno del loop WHILE…DO.



Per arrestare il ciclo prima che le condizioni diventino vere, usare un'istruzione EXIT.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
il ciclo del costrutto è troppo lungo	6	1

Esempio 1:

Per	Immettere questo testo strutturato

Il ciclo WHILE...DO valuta prima le proprie condizioni. Se le condizioni sono vere, il controllore esegue le istruzioni all'interno del ciclo.
 	
Vi è una differenza rispetto al ciclo REPEAT...UNTIL, dato che il ciclo REPEAT...UNTIL esegue le istruzioni nel costrutto e quindi stabilisce se le condizioni sono vere prima di eseguire di nuovo le istruzioni. Le istruzioni di un ciclo REPEAT...UNTIL vengono sempre eseguite almeno una volta. Le istruzioni di un ciclo WHILE...DO potrebbero non essere mai eseguite.
	`pos := 0;`
	`While ((pos <= 100) & structarray[pos].value <> targetvalue)) do`
	`  pos := pos + 2;`
	`  String_tag.DATA[pos] := SINT_array[pos];`
	`end_while;`

Esempio 2:

Per	Immettere questo testo strutturato

Spostare caratteri ASCII da una matrice SINT a un tag stringa (in una matrice SINT ogni elemento contiene un carattere.) Arrestare quando si raggiunge il ritorno a capo.
 	
1. Inizializzare *Element_number* a 0.
 	
2. Contare il numero di elementi in *SINT_array* (matrice contenente i caratteri ASCII) e memorizzare il risultato in *SINT_array_size* (tag DINT).
 	
3. Se il carattere a *SINT_array[element_number]* = 13 (valore decimale del ritorno a capo), arrestare.
 	
4. Impostare *String_tag[element_number]* = il carattere a *SINT_array[element_number].*
 	
5. Aggiungere 1 a *element_number.* In questo modo, il controllore verifica il carattere successivo in *SINT_array.*
 	
6. Impostare il membro Length di *String_tag* = *element_number.* Viene così memorizzato il numero di caratteri fino a quel momento in *String_tag*.
 	
7. Se *element_number* = *SINT_array_size,* arrestare. Si è alla fine della matrice e questa non contiene un ritorno a capo.
 	
8. Andare al 3.
	`element_number := 0;`
	`SIZE(SINT_array, 0, SINT_array_size);`
	`While SINT_array[element_number] <> 13 do`
	`  String_tag.DATA[element_number] := SINT_array[element_number];`
	`  element_number := element_number + 1;`
	`  String_tag.LEN := element_number;`
	`  If element_number = SINT_array_size then`
	`  exit;`
	`  end_if;`
	`end_while;`

REPEAT…UNTIL

Utilizzare il ciclo REPEAT…UNTIL per continuare a eseguire un'azione finché le condizioni non diventano vere.

Operandi:

Testo strutturato

```
REPEAT
    <istruzione>;
UNTIL espressione_booleana
END_REPEAT;
```

Operando	Tipo	Formato	Digitare
espressione_booleana	BOOL	tag espressione	Tag booleano o espressione booleana che restituisce un valore booleano (espressione booleana)

IMPORTANTE

Non creare un numero eccessivo di iterazioni all'interno del ciclo in una singola scansione.

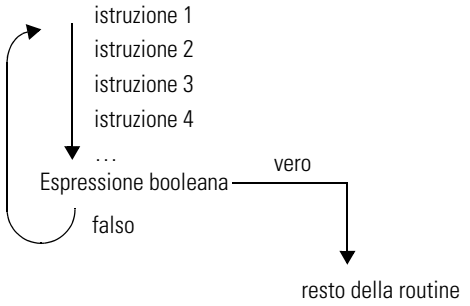
- Il controllore *non* esegue altre istruzioni della routine finché non completa il ciclo.
- Se il tempo necessario per completare il ciclo è maggiore del timer di watchdog per l'attività, si verificherà un errore grave.
- Provare a utilizzare un costrutto diverso, ad esempio IF...THEN.

Descrizione: La sintassi è:

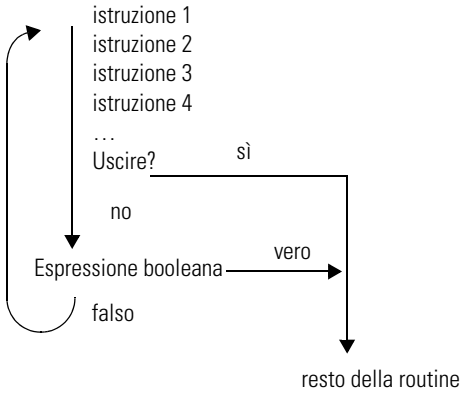
```
REPEAT
   <istruzione>;          ◄── istruzioni da eseguire mentre espressione_booleana1 è falsa

   IF espressione_booleana2 THEN
     EXIT;                ◄── Se vi sono condizioni in cui si desidera uscire anticipatamente dal ciclo, usare altre istruzioni, ad esempio un costrutto IF...THEN, per condizionare un'istruzione EXIT.
   END_IF;

UNTIL espressione_booleana1
END_REPEAT;
```

facoltativo

I seguenti diagrammi mostrano come viene eseguito il ciclo REPEAT...UNTIL e come un'istruzione EXIT produce un'uscita anticipata dal ciclo.



Mentre *espressione_booleana* è falsa, il controllore esegue solo le istruzioni all'interno del ciclo REPEAT…UNTIL.



Per arrestare il ciclo prima che le condizioni diventino false, usare un'istruzione EXIT.

Indicatori di stato aritmetico: non influenzati

Condizioni di errore:

Si verifica un errore grave se	Tipo errore	Codice errore
il ciclo del costrutto è troppo lungo	6	1

Esempio 1:

Per	Immettere questo testo strutturato

Il ciclo REPEAT...UNTIL esegue le istruzioni del costrutto e quindi stabilisce se le condizioni sono vere prima di eseguire di nuovo le istruzioni.
 	
Vi è una differenza rispetto al ciclo WHILE...DO, dato che il ciclo WHILE...DO valuta prima le proprie condizioni. Se le condizioni sono vere, il controllore esegue le istruzioni all'interno del ciclo. Le istruzioni di un ciclo REPEAT...UNTIL vengono sempre eseguite almeno una volta. Le istruzioni di un ciclo WHILE...DO potrebbero non essere mai eseguite.
	`pos := –1;`
	`REPEAT`
	`  pos := pos + 2;`
	`UNTIL ((pos = 101) OR (structarray[pos].value = targetvalue))`
	`end_repeat;`

Esempio 2:

Per	Immettere questo testo strutturato

Spostare caratteri ASCII da una matrice SINT a un tag stringa (in una matrice SINT ogni elemento contiene un carattere.) Arrestare quando si raggiunge il ritorno a capo.
 	
1. Inizializzare *Element_number* a 0.
 	
2. Contare il numero di elementi in *SINT_array* (matrice contenente i caratteri ASCII) e memorizzare il risultato in *SINT_array_size* (tag DINT).
 	
3. Impostare *String_tag[element_number]* = il carattere a *SINT_array[element_number].*
 	
4. Aggiungere 1 a *element_number.* In questo modo, il controllore verifica il carattere successivo in *SINT_array.*
 	
5. Impostare il membro Length di *String_tag* = *element_number.* Viene così memorizzato il numero di caratteri fino a quel momento in *String_tag*.
 	
6. Se *element_number* = *SINT_array_size,* arrestare. Si è alla fine della matrice e questa non contiene un ritorno a capo.
 	
7. Se il carattere a *SINT_array[element_number]* = 13 (valore decimale del ritorno a capo), arrestare.
 	
Altrimenti, andare al 3.
	`element_number := 0;`
	`SIZE(SINT_array, 0, SINT_array_size);`
	`Repeat`
	`  String_tag.DATA[element_number] := SINT_array[element_number];`
	`  element_number := element_number + 1;`
	`  String_tag.LEN := element_number;`
	`  If element_number = SINT_array_size then`
	`  exit;`
	`  end_if;`
	`Until SINT_array[element_number] = 13`
	`end_repeat;`

Commenti

Usare i commenti per semplificare l'interpretazione del testo strutturato.

- I commenti consentono di usare linguaggio naturale per descrivere il funzionamento del testo strutturato.
- I commenti non influenzano l'esecuzione del testo strutturato.

Per aggiungere commenti al testo strutturato:

Per aggiungere un commento	Utilizzare uno di questi formati
su una riga	`//commento`
alla fine di una riga di testo strutturato	`(*commento*)` `/*commento*/`
in una riga di testo strutturato	`(*commento*)` `/*commento*/`
su più righe	`(*inizio commento . . . fine commento*)` `/*inizio commento . . . fine commento*/`

Ad esempio:

Formato	Esempio

//commento	**All'inizio di una riga**
	`//Verifica direzione nastro trasportatore`
	`IF direzione_nastro THEN...`
	
	Alla fine di una riga
	`ELSE //Se nastro fermo, imposta spia allarme`
	`spia := 1;`
	`END_IF;`

(**commento**)	`Apertura.Zucchero[:=]1;(*apri apertura*)`
	
	`IF Zucchero.Basso(*livello minimo LS*)&`
	`Zucchero.Alto(*livello massimo LS*)THEN...`
	
	`(*Controlla la velocità della pompa di ricircolo.`
	`La velocità dipende dalla temperatura nel serbatoio.*)`
	`IF temp.serbatoio> 200 THEN...`

/**commento*/	`Apertura.Zucchero[:=]0;(*chiudi apertura*)`
	
	`IF codice_barre=65 /*A*/ THEN...`
	
	`/*Ottiene il numero di elementi nella matrice`
	`inventario e memorizza il valore nel tag`
	`Voci_Inventario*/`
	`SIZE(Inventorio,0,Voci_Inventario);`

A

abilitazione interrupt utente 448
addizione 250
aggancio dati 634
allarme digitale 30
allarmi 496
AND booleano 314
AND di bit 302
arcocoseno 524
arcoseno 521
arcotangente 527
aritmetica e logica di file 331
ASCII
  assegnazione di testo strutturato 655
assegnazione
  carattere ASCII 655
  non ritentiva 654
  ritentiva 653
assume data available 637, 638, 639
attributi
  conversione di tipi di dati 627
  valori immediati 627
attributi comuni 627
  conversione di tipi di dati 627
  valori immediati 627
aziona task evento 458
azzera 296

B

BAND 314
banda morta 506
bias dell'uscita 507
BNOT 323
BOR 317
break 467
BXOR 320

C

cache
  connessione 170
calcolo 246
cancella buffer ASCII 567
caratteri ASCII in buffer 565
carica FIFO 388
carica LIFO 400
carica sequenziatore 422
CASE 667
codici di errore
  istruzione MSG 150
codici errore
  ASCII 560
codici prodotto 178
commenti
  testo strutturato 679
compensazione anticipata 507
concatenazione stringhe 593
condizioni di overflow 640
configurazione 156
  istruzione MSG 156
  istruzione PID 494
confronto 204
confronto bit di file 472
connessione
  cache 170
connettore
  diagramma a blocchi funzione 633
connettore filo ingresso 633
connettore filo uscita 633
contare in modo decrementale 130
contare in modo incrementale 126
controllo Programma/Operatore
  panoramica 646
conversione di tipi di dati 627
conversione in BCD 548
conversione in interi 551
conversione in scala 497
copiare 350
copiatura sincrona 350
coseno 515
costrutto
  testo strutturato 663
count up/down 134

D

da DINT a stringa 611
da REAL a stringa 613
da stringa A DINT 606
da stringa a REAL 608
descrizione
  testo strutturato 679
deviazione standard 370
diagramma a blocchi funzione
  creazione di un ritardo di scansione 639
  risoluzione del flusso di dati tra blocchi 638
  risoluzione di un loop 637
  selezione elementi 633
dimensione in elementi 375
disabilitazione interrupt utente 448
distribuisci campo di bit 290
distribuisci campo di bit con Target 293
diverso 240
divisione 259
divisione modulo 264
documento
  testo strutturato 679

E

elementi
istruzione SIZE 375
eliminare stringhe 595
esecuzione, ordine 636
esponenziale 538
espressione
espressione booleana
testo strutturato 655
espressione numerica
testo strutturato 655
ordine di esecuzione
testo strutturato 661
testo strutturato
funzione 657
operatori aritmetici 657
operatori bit a bit 661
operatori logici 660
operatori relazionali 658
panoramica 655
espressione booleana
testo strutturato 655
espressione numerica 655
espressioni
formato 206, 248, 341, 347
operatori validi 206, 248, 340, 347
ordine delle operazioni 207, 249, 341, 348
etichetta 428, 619, 623
examine if open 78

F

fine temporanea 444
FOR…DO 670
funzione
testo strutturato 657

G

gradi 542

I

ICON 633
IF…THEN 664
imposta valore di sistema 173
indicatori di stato aritmetico
overflow 640
inserisci stringhe 599
IREF 633
istruzione ABL 562
istruzione ABS 275
istruzione ACB 565
istruzione ACL 567
istruzione ACS 524
istruzione ADD 250
istruzione AFI 450
istruzione AHL 569
istruzione ALMA 42
istruzione ALMD, istruzioni di allarmi ed eventi
ALMD 30
istruzione AND 302
istruzione ARD 573
istruzione ARL 577
istruzione ASN 521
istruzione ATN 527
istruzione AVE 360
istruzione AWA 581
istruzione AWT 586
istruzione BRK 467
istruzione BSL 380
istruzione BSR 384
istruzione BTD 290
istruzione BTDT 293
istruzione CLR 296
istruzione CMP 204
istruzione CONCAT 593
istruzione COP 350
istruzione COS 515
istruzione CPS 350
istruzione CPT 246
istruzione CTU 126
istruzione CTUD 134
istruzione DDT
modalità di ricerca 481
operandi 479
istruzione DEG 542
istruzione DELETE 595
istruzione di attivazione task evento 458
istruzione di fine transizione 452
istruzione di uscita immediata 198
istruzione DIV 259
istruzione DTOS 611
istruzione DTR 486
istruzione EOT 452
istruzione EQU 209
istruzione EVENT 458
istruzione FAL
modalità operativa 326
operandi 331
istruzione FBC
modalità di ricerca 474
operandi 472
istruzione FFL 388
istruzione FFU 394
istruzione FIND 597
istruzione FLL 356
istruzione FOR 464
istruzione FRD 551
istruzione FSC
modalità operativa 326
operandi 342

istruzione GEQ 213
istruzione GRT 217
istruzione GSV
oggetti 176
operandi 173
istruzione INSERT 599
istruzione IOT 198
Istruzione JMP 428
istruzione JMP 619, 623
istruzione JSR 430
istruzione JXR
struttura di controllo 442
istruzione LBL 428, 619, 623
istruzione LEQ 221
istruzione LES 225
istruzione LFL 400
istruzione LFU 406
istruzione LIM 229
istruzione LN 532
istruzione LOG 535
istruzione LOWER 617
istruzione MCR 446
istruzione MEQ 235
istruzione MID 601
istruzione MOD 264
Istruzione MOV 281
istruzione MSG 156
codici di errore 150
connessione con cache 170
metodo di comunicazione 169
operandi 142
regole di programmazione 172
struttura 142
istruzione MUL 256
Istruzione MVM 283
Istruzione MVMT 286
istruzione NEG 272
istruzione NOP 451
istruzione NOT 311
istruzione ONS 86
istruzione OR 305
istruzione OSF 92
istruzione OSFI 97
istruzione OSR 89
istruzione OSRI 94
istruzione OTE 80
istruzione OTL 82
istruzione OTU 84
istruzione pausa SFC 454
istruzione PID
allarmi 496
banda morta 506
bias dell'uscita 507
compensazione anticipata 507
configurazione 494
conversione in scala 497
messa a punto 495
operandi 489
istruzione RAD 545
istruzione RES 139
istruzione RET 430, 468
istruzione ripristina SFC 456
istruzione RTO 110
istruzione RTOR 122
istruzione RTOS 613
istruzione SBR 430
istruzione sempre falso 450
istruzione SFP 454
istruzione SFR 456
istruzione SIN 512
istruzione SIZE 375
istruzione SQI 414
istruzione SQL 422
istruzione SQO 418
istruzione SQR 268
istruzione SRT 365
istruzione SSV
oggetti 176
operandi 173
istruzione STOD 606
istruzione STOR 608
istruzione SUB 253
Istruzione SWPB 298
istruzione TAN 518
istruzione TND 444
istruzione TOD 548
istruzione TOF 106
istruzione TOFR 118
istruzione TON 102
istruzione TONR 114
istruzione TRN 553
istruzione UID 448
istruzione UIE 448
istruzione UPPER 615
istruzione XIO 78
istruzione XOR 308
istruzione XPY 538
istruzioni
allarmi ed eventi 29
bit 75
calcolo 245
confronto 203
contatore 101
controllo programma 427
conversione 541
conversione ASCII 603
conversione matematica 541
conversione stringhe 603
debug 619
for/break 463
input/output 141
logiche 279
manipolazione stringhe 591
manipolazione stringhe ASCII 591

matrice movimento 279
porta seriale 557
porta seriale ASCII 557
scorrimento 379
sequenziatore 413
speciali 471
timer 101
trigonometriche 511
istruzioni ASCII
ABL 562
ACB 565
ACL 567
AHL 569
ARD 573
ARL 577
AWA 581
AWT 586
CONCAT 593
DELETE 595
DTOS 611
FIND 597
INSERT 599
LOWER 617
MID 601
RTOS 613
STOD 606
STOR 608
SWPB 298
UPPER 615
istruzioni CTD 130
istruzioni di allarmi ed eventi
accesso programmatico 68
ALMA, allarme analogico 42
buffer degli allarmi 67
configurazione 61
esecuzione degli allarmi del controllore 71
soppressione o disabilitazione degli allarmi 70
stato allarme 67
testo di un messaggio 64
istruzioni di bit
introduzione 75
ONS 86
OSF 92
OSFI 97
OSR 89
OSRI 94
OTE 80
OTL 82
OTU 84
XIO 78
istruzioni di calcolo
ABS 275
ADD 250
CPT 246
DIV 259
formato dell'espressione 248, 341
introduzione 245
MOD 264
MUL 256
NEG 272
operatori validi 248, 340
ordine delle operazioni 249, 341
SQR 268
SUB 253
istruzioni di confronto
CMP 204
EQU 209
formato dell'espressione 206, 347
GEQ 213
GRT 217
introduzione 203
LEQ 221
LES 225
LIM 229
MEQ 235
NEQ 240
operatori validi 206, 347
ordine delle operazioni 207, 348
istruzioni di contatori
CTD 130
CTU 126
CTUD 134
introduzione 101
RES 139
istruzioni di controllo programma
AFI 450
EOT 452
evento 458
introduzione 427
JMP 428, 619, 623
JSR 430
LBL 428, 619, 623
MCR 446
NOP 451
RET 430
SBR 430
TND 444
UID 448
UIE 448
istruzioni di conversione
DEG 542
FRD 551
introduzione 541
RAD 545
TOD 548
TRN 553

istruzioni di conversione matematica
DEG 542
FRD 551
introduzione 541
RAD 545
TOD 548
TRN 553
istruzioni di conversione stringhe
DTOS 611
introduzione 603
LOWER 617
RTOS 613
STOD 606
STOR 608
SWPB 298
UPPER 615
istruzioni di debug 619
istruzioni di I/O
GSV 173
introduzione 141
IOT 198
MSG 142
SSV 173
istruzioni di manipolazione stringhe
CONCAT 593
DELETE 595
FIND 597
INSERT 599
introduzione 591
MID 601
istruzioni di matematica avanzata
introduzione 531
LN 532
LOG 535
XPY 538
istruzioni di matrice
AVE 360
BSL 380
BSR 384
COP 350
CPS 350
DDT 479
FAL 331
FBC 472
FFL 388
FFU 394
file/varie 325
FLL 356
FSC 342
LFL 400
LFU 406
modalità operativa 326
RES 139
scorrimento 379
sequenziatore 413
SIZE 375
SQI 414
SQL 422
SQO 418
SRT 365
STD 370
istruzioni di movimento
BTD 290
BTDT 293
CLR 296
introduzione 279
MOV 281
MVM 283
MVMT 286
istruzioni di movimento/logiche
BAND 314
BNOT 323
BOR 317
BXOR 320
istruzioni di scorrimento
BSL 380
BSR 384
FFL 388
FFU 394
introduzione 379
LFL 400
LFU 406
istruzioni di timer
introduzione 101
RES 139
RTO 110
RTOR 122
TOF 106
TOFR 118
TON 102
TONR 114
istruzioni file. Vedere istruzioni di matrici
istruzioni for/break
BRK 467
FOR 464
introduzione 463
RET 468
istruzioni logiche
AND 302
introduzione 279
NOT 311
OR 305
XOR 308

istruzioni NEQ 240
istruzioni porta seriale
ABL 562
ACB 565
ACL 567
AHL 569
ARD 573
ARL 577
AWA 581
AWT 586
introduzione 557
istruzioni sequenziatore
introduzione 413
SQI 414
SQL 422
SQO 418
istruzioni speciali
DDT 479
DTR 486
FBC 472
introduzione 471
PID 489
SFP 454
SFR 456
istruzioni STD 370
istruzioni trigonometriche
ACS 524
ASN 521
ATN 527
COS 515
introduzione 511
SIN 512
TAN 518

L

lettura ASCII 573
lettura linea ASCII 577
limite 229
LOG
base 10 535
naturale 532
logaritmo in base 10 535
logaritmo naturale 532
loop di feedback
diagramma a blocchi funzione 637
loop non risolto
diagramma a blocchi funzione 637
lower case 617

M

maggiore di 217
maggiore di o uguale a 213
maschere 487
media 360
messa a punto 495
messaggio 142
connessioni cache 170
regole di programmazione 172
minore di 225
minore di o uguale a 221
modalità continua 326
modalità di ricerca 474, 481
modalità di temporizzazione 641
modalità discreta 327
modalità incrementale 329
modalità operativa 326
moltiplicazione 256
movimento 281

N

negazione 272
nessuna operazione 451
NOT booleano 323
NOT di bit 311

O

OCON 633
oggetti
CONTROLLER 177
CONTROLLERDEVICE 178
CST 179
DF1 181
FAULTLOG 184
istruzione GSV/SSV 176
MESSAGE 185
MODULE 187
MOTIONGROUP 188
PROGRAM 189
ROUTINE 190
SERIALPORT 190
TASK 192
WALLCLOCKTIME 194
oggetto CONTROLLER 177
oggetto CONTROLLERDEVICE 178
oggetto CST 179
oggetto DF1 181
oggetto FAULTLOG 184
oggetto MESSAGE 185
oggetto MODULE 187

oggetto MOTIONGROUP 188
oggetto PROGRAM 189
oggetto ROUTINE 190
oggetto SERIALPORT 190
oggetto TASK 192
oggetto WALLCLOCKTIME 194
one shot 86
one shot falling 92
one shot falling with input 97
one shot rising 89
one shot rising with input 94
operatore 206, 248, 340, 347
  ordine di esecuzione
    testo strutturato 661
operatori aritmetici
  testo strutturato 657
operatori bit a bit
  testo strutturato 661
operatori logici
  testo strutturato 660
operatori matematici
  testo strutturato 657
operatori relazionali
  testo strutturato 658
OR booleano 317
OR di bit 305
OR di bit esclusivo 308
OR esclusivo booleano 320
ordina 365
ordine delle operazioni 207, 249, 341, 348
ordine di esecuzione 636
  espressione di testo strutturato 661
OREF 633
ottieni valore di sistema 173
output energize 80
output latch 82
output unlatch 84

P

post-scansione
  testo strutturato 654
proporzionale, integrale e derivativo 489

R

radianti 545
radice quadrata 268
REPEAT…UNTIL 676
reset 139
reset controllo master 446
retentive timer on 110
retentive timer on with reset 122
ricerca e confronta file 342
ricerca stringhe 597
riempi file 356
riferimento ingresso 633
riferimento uscita 633
righe handshake ASCII 569
rileva diagnostica 479
ritardo scansione
  diagramma a blocchi funzione 639
ritorno 430, 468

S

salta a subroutine 430
salto 428, 619, 623
scarica FIFO 394
scarica LIFO 406
scrittura ASCII 586
scrittura con aggiunta ASCII 581
seno 512
sequenziatore di ingresso 414
sequenziatore di uscita 418
sottrazione 253
sovrapposizione 192
sposta bit a destra 384
sposta bit a sinistra 380
spostamento con maschera 283
spostamento mascherato con target 286
status
  TASK 192
stringa
  valutazione nel testo strutturato 659
stringa intermedia 601
struttura COMPARE 473, 480
struttura CONTROL 332, 380, 384, 389, 395, 400, 401, 407
struttura COUNTER 126, 130
struttura di CONTROL 342
struttura di controllo 361, 365, 370, 414, 418, 422, 442
struttura FBD_COMPARE 210, 214, 218, 222, 226, 241
struttura FBD_LIMIT 230
struttura FBD_BIT_FIELD_DISTRIBUTE 293
struttura FBD_BOOLEAN_AND 314
struttura FBD_BOOLEAN_NOT 323
struttura FBD_BOOLEAN_OR 317
struttura FBD_BOOLEAN_XOR 320
struttura FBD_CONVERT 548, 551
struttura FBD_COUNTER 134
struttura FBD_LOGICAL 303, 306, 309, 312
struttura FBD_MASK_EQUAL 236
struttura FBD_MASKED_MOVE 286
struttura FBD_MATH 251, 254, 257, 260, 265, 273, 539
struttura FBD_MATH_ADVANCED 269, 276, 512, 516, 519, 521, 524, 527, 532, 536, 542, 545

struttura FBD_ONESHOT 94, 97
struttura FBD_TIMER 114, 118, 122
struttura FBD_TRUNCATE 553
struttura messaggio 142
struttura PID 490
struttura RESULT 473, 480
struttura SERIAL_PORT_CONTROL 558, 560, 562, 565, 570, 574, 578, 582, 587
struttura STRINGA 559, 592, 605
struttura TIMER 102, 106, 110
strutture
COMPARE 473, 480
CONTROL 332, 342, 361, 365, 370, 380, 384, 389, 395, 400, 401, 407, 414, 418, 422
COUNTER 126, 130
FBD_BIT_FIELD_DISTRIBUTE 293
FBD_BOOLEAN_AND 314
FBD_BOOLEAN_NOT 323
FBD_BOOLEAN_OR 317
FBD_BOOLEAN_XOR 320
FBD_COMPARE 210, 214, 218, 222, 226, 241
FBD_CONVERT 548, 551
FBD_COUNTER 134
FBD_LIMIT 230
FBD_LOGICAL 303, 306, 309, 312
FBD_MASKED_MOVE 286
FBD_MASK_EQUAL 236
FBD_MATH 251, 254, 257, 260, 265, 273, 539
FBD_MATH_ADVANCED 269, 276, 512, 516, 519, 521, 524, 527, 532, 536, 542, 545
FBD_ONESHOT 94, 97
FBD_TIMER 114, 118, 122
FBD_TRUNCATE 553
istruzione RES 139
MESSAGE 142
PID 490
RESULT 473, 480
SERIAL_PORT_CONTROL 558, 560, 562, 565, 570, 574, 578, 582, 587
STRING 559, 592, 605
stringa 559, 592, 605
TIMER 102, 106, 110
subroutine 430
swap byte 298

T

tangente 518
TASK
attivazione tramite tag consumato 198
aziona task evento 458
configurazione programmatica 192
monitoraggio 192
task evento
attivazione tramite istruzione EVENT 458
attivazione tramite tag consumato 198
configurazione 192
test ASCII per linea buffer 562
testo strutturato
assegnare un carattere ASCII 655
assegnazione 653
assegnazione non ritentiva 654
CASE 667
commenti 679
componenti 651
costrutti 663
espressione 655
espressione numerica 655
FOR…DO 670
funzione 657
IF...THEN 664
operatori aritmetici 657
operatori bit a bit 661
operatori logici 660
operatori relazionali 658
REPEAT…UNTIL 676
valutazione delle stringhe 659
WHILE…DO 673
timeout
configurazione di task evento 192
timer off delay 106
timer off delay with reset 118
timer on delay 102
timer on delay with reset 114
tipo di dati stringa 559, 592, 605
transizione dati 486
tronca 553

U

uguale a 209
uguale mascherato 235
upper case 615
uscita
abilitazione o disabilitazione dell'elaborazione alla fine del task 192
aggiornamento immediato 198
uscita immediata 198
uso di tipi di dati misti 627

V

valore assoluto 275
valori immediati 627

W

WHILE…DO 673

X

X alla potenza di Y 538

Codici dei caratteri ASCII

Carattere	Dec.	Esad.	Carattere	Dec.	Esad.	Carattere	Dec.	Esad.	Carattere	Dec.	Esad.
[ctrl-@] NUL	0	$00	SPACE	32	$20	@	64	$40	‘	96	$60
[ctrl-A] SOH	1	$01	!	33	$21	a	65	$41	a	97	$61
[ctrl-B] STX	2	$02	“	34	$22	B	66	$42	b	98	$62
[ctrl-C] ETX	3	$03	#	35	$23	c	67	$43	c	99	$63
[ctrl-D] EOT	4	$04	$	36	$24	D	68	$44	d	100	$64
[ctrl-E] ENQ	5	$05	%	37	$25	e	69	$45	e	101	$65
[ctrl-F] ACK	6	$06	&	38	$26	f	70	$46	f	102	$66
[ctrl-G] BEL	7	$07	‘	39	$27	g	71	$47	g	103	$67
[ctrl-H] BS	8	$08	(	40	$28	h	72	$48	h	104	$68
[ctrl-I] HT	9	$09	)	41	$29	i	73	$49	i	105	$69
[ctrl-J] LF	10	$I ($0A)	*	42	$2A	j	74	$4A	j	106	$6A
[ctrl-K] VT	11	$0B	+	43	$2B	k	75	$4B	k	107	$6B
[ctrl-L] FF	12	$0C	,	44	$2C	l	76	$4C	l	108	$6C
[ctrl-M] CR	13	$r ($0D)	-	45	$2D	m	77	$4D	m	109	$6D
[ctrl-N] SO	14	$0E	.	46	$2E	n	78	$4E	n	110	$6E
[ctrl-O] SI	15	$0F	/	47	$2F	o	79	$4F	o	111	$6F
[ctrl-P] DLE	16	$10	0	48	$30	p	80	$50	p	112	$70
[ctrl-Q] DC1	17	$11	1	49	$31	q	81	$51	q	113	$71
[ctrl-R] DC2	18	$12	2	50	$32	r	82	$52	r	114	$72
[ctrl-S] DC3	19	$13	3	51	$33	s	83	$53	s	115	$73
[ctrl-T] DC4	20	$14	4	52	$34	T	84	$54	t	116	$74
[ctrl-U] NAK	21	$15	5	53	$35	u	85	$55	u	117	$75
[ctrl-V] SYN	22	$16	6	54	$36	v	86	$56	v	118	$76
[ctrl-W] ETB	23	$17	7	55	$37	w	87	$57	w	119	$77
[ctrl-X] CAN	24	$18	8	56	$38	X	88	$58	X	120	$78
[ctrl-Y] EM	25	$19	9	57	$39	y	89	$59	y	121	$79
[ctrl-Z] SUB	26	$1A	:	58	$3A	z	90	$5A	z	122	$7A
ctrl-[ESC	27	$1B	;	59	$3B	[	91	$5B	{	123	$7B
[ctrl-\] FS	28	$1C	<	60	$3C	\	92	$5C	|	124	$7C
ctrl-] GS	29	$1D	=	61	$3D	]	93	$5D	}	125	$7D
[ctrl-^] RS	30	$1E	>	62	$3E	^	94	$5E	~	126	$7E
[ctrl-_] US	31	$1F	?	63	$3F	_	95	$5F	DEL	127	$7F

Assistenza Rockwell Automation

Rockwell Automation fornisce informazioni tecniche sul web per assistere i clienti nell'utilizzo dei prodotti. Collegandosi al sito http://support.rockwellautomation.com, è possibile consultare manuali tecnici, una knowledge base di Domande frequenti, note tecniche e applicative, codici campione e collegamenti ai service pack del software nonché utilizzare la funzione MySupport personalizzabile per ottimizzare l'utilizzo di questi strumenti.

Per ricevere ulteriore assistenza tecnica telefonica per l'installazione, la configurazione e la ricerca guasti, sono disponibili i programmi di assistenza TechConnect. Per maggiori informazioni, rivolgersi al distributore o al rappresentante Rockwell Automation di zona, oppure consultare il sito web http://support.rockwellautomation.com.

Assistenza per l'installazione

Se si riscontra un problema relativo ad un modulo hardware entro le prime 24 ore dall'installazione, si prega di consultare le informazioni contenute in questo manuale. Per ottenere assistenza per la configurazione e la messa in servizio del modulo, è inoltre possibile contattare l'Assistenza Clienti:

Stati Uniti	1.440.646.3223 Lunedì – Venerdì, 8–17 Ora della costa orientale
Fuori dagli Stati Uniti	Per qualsiasi problema di assistenza tecnica, si prega di contattare il rappresentante Rockwell Automation di zona.

Restituzione di prodotti nuovi non funzionanti

Rockwell collauda tutti i prodotti per garantirne la completa funzionalità al momento della spedizione dall'impianto di produzione. Tuttavia, nel caso il prodotto non fosse funzionante e dovesse essere restituito, procedere come segue:

Stati Uniti	Rivolgersi al proprio distributore. Per completare la procedura di reso è necessario fornire al distributore il numero di pratica attribuito dall'Assistenza Clienti (chiamare il numero telefonico sopra indicato per ottenerne uno).
Fuori dagli Stati Uniti	Per la procedura di reso, si prega di contattare il rappresentante Rockwell Automation di zona.

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americhe: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: +1 414 382 2000, Fax: +1 414 382 4444
Europa/Medio Oriente/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Bruxelles, Belgio, Tel: +32 2 663 0600, Fax: +32 2 663 0640
Asia: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: +852 2887 4788, Fax: +852 2508 1846

Italia: Rockwell Automation S.r.l., Via Gallarate 215, 20151 Milano, Tel: +39 02 334471, Fax: +39 02 33447701, www.rockwellautomation.it
Svizzera: Rockwell Automation AG, Buchserstrasse 7, CH-5001 Aarau, Tel.: +41 (62) 889 77 77, Fax: +41 (62) 889 77 11

Pubblicazione 1756-RM003K-IT-P – Luglio 2008
Sostituisce la pubblicazione 1756-RM003I-IT-P – Gennaio 2007
Copyright © 2008 Rockwell Automation, Inc. Tutti i diritti riservati. Stampato negli U.S.A.

Allen-Bradley

Titolo Pubblicazione

Tipo di pubblicazione